TECHNISCHE UNIVERSITÄT
BERGAKADEMIE FREIBERG
The University of Resources. Since 1765.

Behavior-Specific Proprioception Models for Robotic Force Estimation: A Machine Learning Approach

Von der Fakultät für Mathematik und Informatik
der Technischen Universität Bergakademie Freiberg

genehmigte

DISSERTATION

zur Erlangung des akademischen Grades

Doktor-Ingenieur

(Dr.-Ing.)

vorgelegt von **M.Sc. Erik Berger**

geboren am 28. Oktober 1985 in Grimma

Gutachter: Prof. Dr.-Ing. habil. Bernhard Jung, Freiberg, Deutschland
Prof. Dr.-Ing. Heni Ben Amor, Tempe, USA

Tag der Verleihung: 5. Juli 2018

Versicherung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts habe ich Unterstützungsleistungen von folgenden Personen erhalten: Die Betreuer der vorliegenden Dissertation, Bernhard Jung und Heni Ben Amor, gaben Hinweise für die Überarbeitung früherer Versionen des Manuskripts. Teile des dargestellten Materials entstanden im Rahmen von mir betreuter studentischer Qualifizierungsarbeiten von Christian Pönisch, David Müller und Daniel Eger Passos. Weitere Personen waren an der Abfassung der vorliegenden Arbeit nicht beteiligt. Die Hilfe eines Promotionsberaters habe ich nicht in Anspruch genommen. Weitere Personen haben von mir keine geldwerten Leistungen für Arbeiten erhalten, die nicht als solche kenntlich gemacht worden sind.

Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Freiberg, den 2. Mai 2018

Erik Berger

Acknowledgement

If a machine is expected to be infallible, it cannot also be intelligent.

—Alan Turing

I would like to thank my advisers Bernhard Jung and Heni Ben Amor for their extraordinary mentorship. Whether via scientific or philosophic debates, they managed to keep me focused for years of research. Preserving the motivation for writing this thesis owes a lot on their encouragement. I would also like to thank my former colleague and friend David Vogt who was willing to support me for more than a decade. His constructive criticism and expertise in graphic design was from immense value. A special thank is addressed to my friend Lars Kafurke who helped distracting me whenever I needed a break.

I am deeply thankful to the support of my amazing wife Mandy who gives me mental strength and energy to achieve my goals. Our beloved children are frequently surprising me with their learning capabilities and inspired quite a lot of my research. Finally, I want to thank my parents, grand-parents, parents-in-law and siblings for their general support and unconditional trust in all situations of life.

Contents

Previous Publications

This thesis contains and extends material from the author's own publications:

[1] Erik Berger, David Vogt, Christian Pönisch, Heni Ben Amor, and Bernhard Jung. *Cooperative human-robot manipulation tasks.* IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Beyond Robot Grasping - Modern Approaches for Learning Dynamic Manipulation, 2012

[2] Erik Berger, David Vogt, Heni Ben Amor, and Bernhard Jung. *Behavior adaptation in cooperative human-robot transportation tasks.* IEEE International Conference on Robotics, Automation (ICRA), Workshop on Semantics, Identification, and Control of Robot-Human-Environment Interaction, 2013

[3] Erik Berger, David Vogt, Nooshin Haji-Ghassemi, Bernhard Jung, and Heni Ben Amor. "Inferring guidance information in cooperative human-robot tasks". In: *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on.* IEEE. 2013, pp. 124–129

[4] Erik Berger, Mark Sastuba, David Vogt, Bernhard Jung, and Heni Ben Amor. "Dynamic mode decomposition for perturbation estimation in human robot interaction". In: *Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on.* IEEE. 2014, pp. 593–600

[5] Erik Berger, David Müller, David Vogt, Bernhard Jung, and Heni Ben Amor. "Transfer entropy for feature extraction in physical human-robot interaction: Detecting perturbations from low-cost sensors". In: *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on.* IEEE. 2014, pp. 829–834

[6] Erik Berger, Mark Sastuba, David Vogt, Bernhard Jung, and Heni Ben Amor. "Estimation of perturbations in robotic behavior using dynamic mode decomposition". In: *Advanced Robotics* 29.5 (2015), pp. 331–343

[7] Erik Berger, Steve Grehl, David Vogt, Bernhard Jung, and Heni Ben Amor. *Learning to estimate forces for safe tool usage.* IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Safety for Human-Robot Interaction in Industrial Settings, 2015

[8] Erik Berger, Steve Grehl, David Vogt, Bernhard Jung, and Heni Ben Amor. "Experience-based torque estimation for an industrial robot". In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on.* IEEE. 2016, pp. 144–149

[9] Erik Berger, David Vogt, Steve Grehl, Bernhard Jung, and Heni Ben Amor. "Estimating perturbations from experience using neural networks and information transfer". In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on.* IEEE. 2016, pp. 176–181

This thesis proposes a novel concept for enhancing robotic behavior with proprioceptive models. These models can be used for force estimation based on standard sensors commonly available on modern robotic systems, without the need for dedicated force sensors. The humanoid robot applications described in Chapter 5 are based on material from the authors own articles [1, 2, 3, 4, 5, 6]. Chapter 6 significantly expands the findings of references [7] and [8] which focus on the augmentation of virtual force sensors to the proprioception of an industrial robot arm. The most recent publication [9] combines multiple proprioception models to extend the functionality of state-of-the-art force-torque sensors and is in the core of Chapter 7.

Mathematical Notation

Symbol	Description
$\mathbf{x}$	A vector.
x_i	The ith element of $\mathbf{x}$.
$\mathbf{X}$	A matrix.
$\mathbf{X}_{i,j}$	The element contained in the ith row and the jth column of $\mathbf{X}$.
$\mathbf{x}(i)$	The ith row vector of $\mathbf{X}$.
$\mathbf{x}[i]$	The ith column vector of $\mathbf{X}$.
X	A scalar.
$\tilde{\mathbf{X}}$	A submatrix of $\mathbf{X}$.
$\mathbf{X}'$	Data projected into a low-dimensional feature space.
$\hat{\mathbf{X}}$	A sequence of runtime data in matrix notation.
$\hat{\mathbf{X}}'$	Runtime data projected into a low-dimensional feature space.
$\mathcal{X}, \mathcal{Y}, \mathcal{Z}$	Various discrete random variables or processes.
$H(\mathcal{X})$	The Shannon entropy of $\mathcal{X}$.
$H(\mathcal{X}, \mathcal{Y})$	The joint entropy of $\mathcal{X}$ and $\mathcal{Y}$.
$H(\mathcal{X}\|\mathcal{Y})$	The conditional entropy of $\mathcal{X}$ given $\mathcal{Y}$.
$I(\mathcal{X}; \mathcal{Y})$	The mutual information between $\mathcal{X}$ and $\mathcal{Y}$.
$I(\mathcal{X}; \mathcal{Y}\|\mathcal{Z})$	The conditional mutual information between $\mathcal{X}$, $\mathcal{Y}$ given $\mathcal{Z}$.
$I(\mathcal{X}; \mathcal{Y}; \mathcal{Z})$	The multivariate mutual information between $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$.
$I(\mathcal{X}+1; \mathcal{Y}\|\mathcal{X})$	The transfer entropy from $\mathcal{Y}$ to the future of $\mathcal{X}$.
$\mathcal{N}$	A set of neurons.
$\{\mathcal{I}, \mathcal{H}, \mathcal{O}\} \in \mathcal{N}$	The input, hidden and output layer.
$\mathcal{C}$	A set of directed connections.
$\mathbf{W}$	The connection weights.
$(\mathcal{N}, \mathcal{C}, \mathbf{W})$	The structure of an artificial neural network.

Acronyms

ANN Artificial Neural Network

BSPM Behavior-Specific Proprioception Model

CEC Constant Error Carrousel

CEE Cross Entropy Error

CMI Conditional Mutual Information

CNN Convolutional Neural Network

CNS Central Nervous System

DMD Dynamic Mode Decomposition

DTW Dynamic Time Warp

EUD Euclidean Distance

F-BSPM Behavior-Specific Proprioception Model in Forward Mode

FNN Feed Forward Neural Network

FT Force-Torque

GDMD General Dynamic Mode Decomposition

GP Gaussian Process

GPR Gaussian Process Regression

I-BSPM Behavior-Specific Proprioception Model in Inverse Mode

LLE Locally Linear Embedding

LSTM Long Short-Term Memory

MAE Mean Absolute Error

MC Manifold Charting

MLP Multi-Layer Perceptron

MMI Multivariate Mutual Information

MRE Mean Relative Error

MSE Mean Squared Error

PC Principal Component

PCA Principal Component Analysis

PFS Phase Feature Space

RMSE Root Mean Square Error

RNN Recurrent Neural Network

SDMD Sparsity-Promoting Dynamic Mode Decomposition

SDTW Subsequence Dynamic Time Warping

SLP Single-Layer Perceptron

SUS System Usability Scale

TCP Tool Center Point

TE Transfer Entropy

TFS Torque Feature Space

V-BSPM Behavior-Specific Proprioception Model in Virtual Mode

List of Figures

List of Tables

1 Introduction

Fig. 1.1: The proposed machine learning approach enables different robot platforms to generate accurate force estimates for physically demanding tasks.

An important vision of research in robotics and artificial intelligence is the development of robot capabilities which allow to flexibly support humans under changing environmental conditions. Examples are robotic assistants that can help to transport heavy objects, utilize custom tools or explore unstructured environments (see Figure 1.1). However, close-contact of this kind requires significant sensing capabilities in order to ensure safe and meaningful physical interactions with humans or the environment. Here, humans often rely on force feedback during physical activities.

In order for a robot to engage in similar activities, it needs to be able to sense external forces. This usually requires special purpose sensors which induce several drawbacks such as increased costs and reduced payload. Furthermore, such sensors often return noisy non-zero readings when the robot executes a motor task. Hence, it is difficult to distinguish regular from perturbed behavior execution based on these sensors only. In contrast, humans are capable to estimate forces by utilizing prior proprioceptive experiences. These estimates can then be used to adapt the behavior in an adequate manner.

This thesis presents a machine learning approach which generates experience-based models of robotic proprioception. During training, the robot executes a behavior while gathering information about the fluctuations of its proprioceptive sensors and the corresponding forces. Sensors which are strongly correlated to these forces are then utilized to learn behavior-specific proprioceptive models. As a result, the approach estimates forces without the need of special purpose sensors. The amount and direction of the detected forces can then be utilized to adapt the robot's behavior by appropriate reaction rules.

1.1 Biological Motivation

> *During the learning of any new skill, sport, or art, it is usually necessary to become familiar with some proprioceptive tasks specific to that activity. Without the appropriate integration of proprioceptive input, an artist would not be able to brush paint onto a canvas without looking at the hand as it moved the brush over the canvas; it would be impossible to drive an automobile because a motorist would not be able to steer or use the foot pedals while looking at the road ahead; a person could not touch type or perform ballet; and people would not even be able to walk without watching where they put their feet.* [10]

Humans are able to recognize and compensate unexpected extrinsic perturbations during behavior execution. For example, performing the same lifting behavior for similar shaped but different weighted objects requires the adaptation of muscle tensions. In contrast, most robots are still explicitly programmed to perform the same behavior over and over again. While this may be sufficient for various industrial settings, real world applications are a much greater challenge. Here, changing environmental conditions can have long-term consequences on a continuously executed behavior which also restricts the complexity of reasonable analytical models.

In contrast, the sense of self or proprioception allows humans to learn and adapt a skill by becoming familiar with the particular task. Here, no analytical model is required to adapt the walking gait to environmental constraints, apply an adequate amount of forces to surroundings or to estimate the weight of a lifted object. Instead, experience about the own body gathered during previous behavior executions is generalized with regard to the actual state of the body.

1.1.1 Learning from Experience

Various species are able to imitate a demonstrated behavior. For example, human children learn walking through the haptic influence of their parents [11] and further try to imitate close to all observed behaviors. Bandura and Walters [12] also found that such kind of *imitation learning* is a crucial factor for the cognitive development of children. Different configurations of body characteristics, environmental conditions and task situations inhibit the behavior from being copied directly. Therefore, applying such kind of imitation learning to humanoid robots [13] usually requires direct task transfer and an additional optimization process. For example, haptic interaction is used by Ben Amor et al. [14] to teach a small sized humanoid robot various complex motor skills, e.g., standing-up and walking. Without the stabilizing assistance of the human interactant a successful execution of the recorded motion is not possible. Hence, inside a physical simulation, the motion is optimized and transferred to the robot afterwards. However, the main advantage of this data-driven method is that it starts from a 'bootstrapped' motion which is already close to an optimal solution.

Similar to this, a being which aims to solve a task for unknown environmental conditions rarely begins from scratch. In most of the cases the individual has an initial idea

Tab. 1.1: Biological proprioceptors compared to robotic sensors with related functionality.

Biological Proprioceptors	**Robotic Proprioceptors**
muscle length	joint angle
muscle tension	joint current
muscle spindle	joint velocity/acceleration
Golgi tendon organ	joint torque
Pacinian corpuscle	force/pressure sensors
vestibular system	inertial measurement unit

which is obtained from previously behavior executions and similar situations. Developing such cognitive capabilities requires to gather experience during body-environment interaction [15]. In order to perform and evaluate such interactions an idea about the sensory-motor integration of the body need to be given. More precisely, this means a mapping from *proprioception* to visual and haptic stimuli.

Proprioception refers to the sense of the (1) relative position of one's own parts of the body and (2) strength of effort being employed in movement [16]. For example, (1) humans are able to locate their limbs without the need of visual feedback and (2) produce the needed strength to hold objects. For this, muscle lengths and tensions are compared to previous experienced sensations which allow estimating the actual position, movement and exerted forces. Following Sherrington [17], proprioception is distinguished from *interoception* and *exteroception*. Exteroception refers to senses that provide information originating outside the body, such as sight and hearing while interoception provides information about the stretching of internal organs and pain.

In humans, proprioceptive receptors or *proprioceptors* are able to recognize compression, traction and elongation of muscles where body movement is usually needed to activate them. These are sensors which gather information about the own body configuration and allow to have a sense of self. The most relevant biological proprioceptors are summarized in the left column of Table 1.1. The *muscle spindles* [18] recognize variations in the muscle length while the so called *Golgi tendon organ* [19] measures changes of tension. Similar to this, the *Pacinian corpuscles* [20] are mechanoreceptors which measure pressure and are primarily contained in the skin. In contrast, the vestibular system provides the main contribution to body orientation and balance which also contributes to the measurement of body movement. This makes it difficult to separate the vestibular information from the proprioception and vice versa. State-of-the-art robotic systems provide access to a rich set of sensor readings. The right column of Table 1.1 summarizes commonly used robotic proprioceptors which are related to the biological ones.

In the real world a motor skill is never executed twice in exactly the same way. One way to accommodate such uncertainties is to get familiar with behavior-specific sensor feedback. Utilizing proprioceptive information, this thesis aims to apply the

benefits of experience based learning to implement adaptive robotic behavior. More precisely, a novel machine learning approach is presented for deriving behavior-specific proprioceptive models from previous behavior executions. During training, the behavior is executed several times with different task parametrizations or may be subjected to different extrinsic perturbations. For example, a pick-and-place operation may be executed with objects of different weights or a tool may be used with different forces being applied to external objects. During execution of each behavior example, the proprioceptors are continuously experienced. The resulting time series of sensor readings serves as training data of the robot's proprioception. This experience is used to estimate the proprioceptive state or the executed action of a robot.

1.1.2 The Reafference Principle

The fundamental idea behind the reafference principle is introduced in the research of Steinbuch about the tactile recognition of objects [21]. Here, the central observation is that humans are capable to identify objects by grasping them while a passive collision with the object is insufficient for accurate object recognition. More precisely, the hypothesis is that the human brain combines knowledge about the executed motion with its consequences on the sensor perception which Steinbuch referred to as 'Bewegidee' [21, p. 30].

This idea is further refined by Charles Bell [22] and his observations of how humans perceive a stable world from the images of the eye. He demonstrates that the brain utilizes information about the active eye movement (the eye muscles) to predict changes in the image received by the retina. By combining both, the human brain is able to stabilize the images. Furthermore, he enhanced the former concept by the idea that copied actuator commands are used to predict changes in the sensor perception. This idea is proven by Helmholtz who utilizes the acquired knowledge to describe the functionality of visual object localization [23].

The reafference principle, introduced by Holst and Mittelstaedt [24], describes the concept of a prediction correction procedure. They also formulate a basic model which is simplified illustrated in Figure 1.2 left. Here, an effector command (efference), e.g., a signal to an actuator or muscle, is received from the Central Nervous System (CNS). This *efference* is duplicated and stored as *efference copy*. Next, the efference generates an action which is executed by the effector. During this action execution, the sensory receptors make measurements referred to as *afference*. Here, the values contained in the afference originate from two different sources: *reafference* and *exafference*. A reafference occurs due to regular *intrinsic* fluctuations which are caused by the execution of the efference and depend on the corresponding action and the properties of the system. In contrast, the exafference determines the influence of the surrounding world and is triggered by extrinsic perturbations.

The main problem is that the system's sensory receptors measure the sum of both and cannot differentiate between them. Instead, the afference is compared with the previously created efference copy. More precisely, the difference between afference and

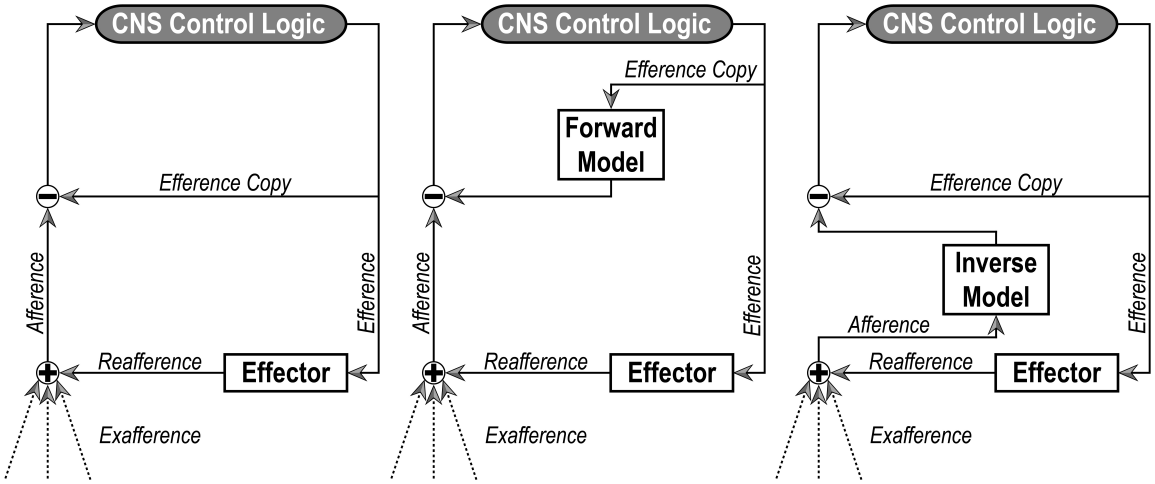


Fig. 1.2: Different processing models for an effector command (efference). Here, an efference is received from the CNS. First, a copy of this efference is created. Next, the effector performs the action contained in the efference resulting in an afference which contains the sensor perception at the effector. Left: In the basic approach, extrinsic perturbations are detected by comparing the efference copy with the afference where a deviation between both cause the control logic contained in the CNS to generate a new efference which is fed back to the circulation. Middle: In contrast, the forward model predicts the afference from the efference copy to allow handling different afference/efference spaces. Right: Similar to this, the inverse model transforms the afference to an estimate of the performed efference.

efference copy determine the exafference. A successful action execution is confirmed when the exafference is zero. Otherwise, a new efference which compensates the exafference is created by the control logic and consequently compensates the corresponding extrinsic perturbations. Broadly speaking, the effector is controlled by comparing the setpoint (efference) with the actual value (afference) in a feedback-loop. This is close to PID controller approaches developed in the field of automation and control technology [25]. Here, the desired setpoint of a process is compared to the actually measured process value where the controller adapts the process based on proportional, integral, and derivative terms in a feedback loop. Therefore, properties such as dimensionality and value-range of the efference motor commands and the sensor afference are assumed to be identical in the basic approach (see Figure 1.2 left).

However, as first taken into account by Roger Wolcott Sperry [26] and his studies about the optokinetic reflex these properties can also differ. In more detail, when following a moving object the efference of the eye muscle generates an afference which is an image perceived at the retina. Hence, efference and afference have different dimensionality and ranges of values. Therefore, motor commands need to be mapped to the corresponding sensor pattern or vice versa. There are two different model-based methods which can be applied to unify the efference and afference spaces: a forward (see Figure 1.2 middle) and an inverse model (see Figure 1.2 right).

A forward model predicts the expected intrinsics from the efference copy and consequently converts motor commands to the particular sensor perception. The difference between the actual perception and the predicted one is used as an estimate of the extrinsic perturbation. This has the advantage that the extrinsic perturbation is measured in the sensor space what therefore allows to implement very specific control rules.

In contrast to such a kind of predictor, the inverse model utilizes the total afference to generate an estimate of the executed efference. The difference between estimated efference and efference copy is used to derive the extrinsic perturbation which therefore is measured in the effector space. Hence, an inverse model is more similar to a controller where extrinsic perturbations correspond to effector commands which can be directly applied to compensate them. Inspired by this, Kawato [27] proposed internal forward and inverse models for the application in robotics. In particular, *forward models* make use of the robot's measured proprioception and the actuator command to predict the expected proprioception while *inverse models* generate an actuator command which causes a desired change in the robot's proprioception.

As mentioned above, some actuator command may have an influence on the particular sensor perception, e.g., image stabilization benefits from actuator commands of the eyes rather than the fingers or arms. An adequate mapping therefore requires knowledge about the sensory-motor integration of the system. More precisely, what consequences does an action performed by the motors have on the perception received by the particular sensor? To solve this question, one could implement an accurate analytical model which takes into account kinematic structure, mass distribution and the dynamics of the system. The complexity of the resulting model and its missing adaptation capabilities actually restricts applications to rigid systems with just a few joints. For example, most of todays robotic arms support direct and inverted kinematics but cannot automatically adapt to changing environmental conditions. Hence, an automatic extraction of such sensory-motor dependencies is discussed in the following.

1.1.3 Sensory-Motor Dependencies

The adequate selection of motor commands and sensors which are relevant to learn the particular task is a core component of the presented machine learning approach. For example, a forward model usually makes use of locally connected muscles which are close to the effector or the particular sensor. Such local correlations are sufficient for a subset of motion sequences but not for more complex tasks which may involve various distal dependencies. To illustrate this, the sense of balance is considered in the following. For humans the main contribution to the prediction of balance is given by the vestibular system in the ears while distal dependencies to the muscles in the eyes, neck, legs, arms and the visual perception are also relevant. A popular example of how the visual perception can disturb the sense of balance is the simulator sickness [28]. Here, the eyes perceive a moving world that cannot be measured by the vestibular sensor system. These contradictory perceptions result in incompatible predictions which are the cause for the sickness.

In this context it is important to note that, in addition to motors (i.e. muscles), also sensors (e.g. visual perception) can contain relevant information and therefore can be used as input to predict other sensors (e.g. balance). Hence, actuator commands and sensor readings can be used as input for the computational model developed in this thesis. Here, the behavior-specific selection of inputs, whether originating from the

actuator or sensory system, is referred to as *sensor selection*. A major advantage of utilizing only the relevant subset of sensors is the decreased complexity of the model. In robotic behaviors, such sensory-motor dependencies might be complex and far from obvious. A common solution is to execute a behavior several times under varying conditions while recording the sensory and motor system. The recorded set of behavior examples represents the training data which is utilized to detect sensory-motor dependencies. In the presented approach, statistical methods such as information theory and dimensionality reduction are applied to extract behavior-specific sensory-motor dependencies.

Furthermore, a major challenge is to distinguish statistical correlations from spurious correlations. Correlations describe a statistical relationship, whether causal or not, between two processes that are beneficial for prediction purposes. In contrast, spurious correlations describe wrong correlations which are inferred by coincidence or unknown background processes and are only correct for a limited number of observations. To avoid learning from spurious correlations, the number and heterogeneity of behavior examples contained in the training data are crucial. A common solution to this problem can be achieved by recording an independent validation data set. The validation data can be used to evaluate the accuracy of the learned model where the usage of spurious correlations usually results in large prediction errors.

1.2 Methodology

This thesis is a study of force-estimation with Behavior-Specific Proprioception Models (BSPMs) which allows robots to flexibly interact with their environment. The biological motivation of the proposed methodology is given by the reafference principle which was introduced above. This concept describes how experience about previous behavior executions enables natural beings to distinguish between self-generated and environmental influences. In order to achieve similar capabilities on a robot, the proposed machine learning approach builds on data-driven methods related to the fields of imitation learning and information theory.

First, a behavior which consists of a sequence of motor commands is repeatedly demonstrated and stored together with available proprioceptive readings. The resulting training data therefore contains behavior-specific information about sensory-motor dependencies. By means of statistical methods, this information helps to uncover simultaneous and time delayed correlations in the behavior of the robot. Sensors with a strong proprioceptive correlation are automatically selected for model learning. Here, machine learning algorithms are applied to infer BSPMs which can be configured in three different modes:

- A Behavior-Specific Proprioception Model in Forward Mode (F-BSPM) predicts the reafference or intrinsic proprioception.
- A Behavior-Specific Proprioception Model in Inverse Mode (I-BSPM) estimates an executed efference or behavior parameter configuration.

- In contrast, a Behavior-Specific Proprioception Model in Virtual Mode (V-BSPM) augments the proprioception by combining available information to a *virtual sensor* which is not addressed in the original formulation of the reafference principle.

For this, machine learning algorithms examined in this thesis include: lazy learners, probabilistic models, classical neural networks and deep learning techniques. Furthermore, different preprocessing methods such as principal component analysis, autoencoders, transfer entropy and mutual information are examined.

As a result, BSPMs enable a robot to distinguish between intrinsic sensor readings from extrinsic perturbations without the need of special purpose sensors. In particular, intrinsic fluctuations are caused by regular behavior execution while extrinsic perturbations are induced by application of external forces. Robotic applications examined in this thesis include:

- Make robotic walking behaviors robust against being pushed or pulled by an adversary or forces exerted by a collaborator during joint transportation tasks.
- Calculate quantitative estimates of forces applied by the interaction with external objects during tool usage.
- Accurately classify an object's weight which is being lifted under rough environmental conditions.
- Implement pick-and-place operations which safely detect the influence of intentional physical human-robot interactions and unintended collisions.

Here, the deviation between predicted and measured proprioception is attributed to the influence of extrinsic perturbations. In the absence of external stimuli, this difference is about zero while fluctuations in the proprioception originate from behavior-specific intrinsics only. As emphasized earlier, human perception such as the sense of balance relies on the combination of various information sources and therefore is less sensitive to errors. This is also a main advantage of the presented BSPM approach. For instance, when affected by noise or even defect, a single special purpose sensor would result in failure. Instead, multiple sensors are combined to one higher level of functionality. Consequently, negative effects which occur partially can be compensated by the combined information and still result in meaningful estimates of extrinsic perturbations.

The estimated amount and direction of extrinsic perturbations is then used to perform an appropriate behavior adaption where the robot's reaction rules depend on the particular field of application. The generalizability of the approach presented in this thesis is shown in a variety of applications involving physically interacting robots. For instance, one application utilizes prediction-correction schemes in physical human-robot collaboration on a humanoid robot platform. Here, the robot iteratively corrects its behavior configuration and consequently compensates extrinsic perturbations. Other applications, implemented on an industrial robotic arm, aim on force feedback for tool-usage, physical human-robot interaction and state classification tasks. The basic idea is that providing a robot with accurate force sensing capabilities enables it to safely interact with human participants and technical equipment.

1.3 Contributions

This thesis investigates how proprioceptive feedback can be used for robotic force estimation using a machine learning approach. In doing so, various conceptual, methodological and application-oriented contributions to the state of the art are presented.

Conceptual Contributions

The main contributions of this thesis are Behavior-Specific Proprioception Models (BSPMs) which generate estimates of the expected proprioception from prior experience. In contrast to classical learning of motion commands, these models enable robots to flexibly interact with their environment by adding expected sensations to the received feedback. Here, safe and meaningful physical interactions require distinguishing self-generated sensations from environmental influences. Motivated by proprioceptive models in humans, different BSPM variants are proposed to equip robots with a proprioceptive sense of self.

In particular, an F-BSPM predicts the behavior-specific intrinsic proprioception. This allows estimating the amount and direction of external forces in the space of the particular proprioceptor for a concrete situation. In contrast, an I-BSPM estimates high-level behavior parameters from the measured proprioception. By comparing this estimate with the executed behavior, extrinsic perturbations can be detected in behavior space. Furthermore, V-BSPMs augment robotic proprioception with virtual sensors. These sensors are generated by combining available but usually hidden information from the robot's proprioceptors. As will be shown, the force estimates of these models significantly increase the interaction capabilities of different robot platforms.

Methodological Contributions

The main methodological contribution is the operationalization of the BSPM approach via data-driven machine learning techniques. For this, behavior-specific proprioceptive experiences are acquired from demonstrations. Here, the wear and tear of the robot and the effort spent by the user is reduced by learning from only a few training examples. To further increase the practical applicability of the proposed approach, no expert knowledge about the particular robot platform, e.g., kinematic chains or mass distribution, is required. Instead, various information-theoretic measurements are applied to automatically identify the most relevant sensor readings for force estimation of a specific behavior. To the best knowledge of the author, this is the first time that such information-theoretic measurements are applied in the context of sensor selection in robotic proprioception.

Furthermore, this thesis compares advantages and disadvantages of various lazy and eager machine learning approaches. The precision of lazy learners such as time series alignment algorithms usually depends on a large number of training examples. This thesis introduces a novel interpolation technique which extracts the underlying dynamics of a behavior by utilizing methods from the field of fluid dynamics. Here, a small

set of training examples is sufficient to accurately interpolate sensor readings for so far unknown situations. In contrast, eager learning has the possibility to generalize outputs for arbitrary inputs. Here, the information-theoretic sensor selection method is crucial for learning efficiency. Furthermore, several techniques reaching from statistical methods over classical neural networks up to modern deep learning techniques are compared.

Application-Oriented Contributions

The BSPM approach is applied to different robot platforms which do not possess dedicated force sensing capabilities to fulfill various physically demanding tasks. Here, proprioceptors, behavior parameter configurations and context descriptions can be selected as learning target of the corresponding proprioceptive model. Hence, the presented machine learning approach in general is applicable to any kind of sequential quantity, i.e., regression for continuous values or classification of discrete labels.

One platform is the humanoid NAO robot where the BSPM approach is applied to make walking robust against extrinsic forces which are triggered by the environment rather than the behavior execution itself. As a result, the biped robot can stabilize its walking gait and follow the guidance of a human interactant.

Furthermore, an industrial robotic arm learns to estimate torque values during the usage of a custom tool and classifies weight even more precisely than a special purpose sensor. During performing a pick-and-place task this robot is also enabled to accurately detect collisions with the environment and a human interactant. The resulting force estimates and interaction capabilities are comparable to the usage of state-of-the-art force-torque sensors.

1.4 Thesis Outline

Chapter 2 presents the fundamental concepts and basic terms of machine learning which are widely used in the field of robotics and computer science. Furthermore, related work on data preprocessing, force estimation and neural networks is introduced.

Chapter 3 introduces the mathematical background for detecting sensory-motor dependencies. For this, measures in the field of information theory, which allow detecting correlations within huge data sets, are examined. In particular, mutual information and transfer entropy are introduced and evaluated on the basis of a simple example. Furthermore, the fundamentals of biologically inspired neural networks to state-of-the-art deep learning technologies are introduced.

Chapter 4 introduces the new concept of BSPMs to equip robots with accurate force estimation capabilities. Here, one goal is to distinguish self-generated forces from perturbations which originate from the surrounding environment.

Chapter 5 gives a comparative depiction of the F-BSPM and I-BSPM approach for the application on a humanoid robot. Here, the goal is to infer guidance information during human-robot collaboration from the robot's integrated sensors. The direction of guidance is then used to adapt the robotic behavior in an adequate manner.

Chapter 6 introduces two applications which implement V-BSPMs to augment the proprioception of an industrial robot arm with accurate force sensing capabilities. Here, force related context descriptions are manually assigned to training examples which are then approximated during runtime. In particular, the robot learns to accurately perceive and distinguish torque values applied to a wrench and to estimate the weight of a self-contained water extraction station.

Chapter 7 presents an application that combines multiple BSPMs to accurately detect extrinsic perturbations from force-torque measurements. To this end, a state-of-the-art force-torque sensor is trained in combination with an industrial robotic arm. During runtime, this special purpose sensor is replaced by its virtual counterpart which further allows to distinguish intrinsic fluctuations from extrinsic perturbations.

Chapter 8 concludes this thesis by highlighting the most important findings and contributions made. Finally, some promising directions for future research are given.

2 Related Work on Machine Learning

In this chapter, the fundamental concepts and basic terms of machine learning are introduced. Next, the benefits of data preprocessing for efficient model learning and runtime application are highlighted. Furthermore, several proprioception based applications for adaptive systems which focus on the use of neural network architectures are discussed. In turn, the general applicability, current research and promising directions of Artificial Neural Networks (ANNs) are given. Finally, it is presented how this thesis extends previous work.

2.1 Machine Learning Fundamentals

The field of machine learning is a combination of different disciplines such as statistics, information theory and optimization. As noted by Carbonell, Michalski, and Mitchell [29] it is usually applied to simulate human learning, explore novel learning techniques or to implement learning capabilities in artificial systems. In this thesis, the term 'machine learning' is utilized in the context of the latter and therefore allows a system to 'learn without being explicitly programmed' [30].

A major advantage of machine learning is its general applicability across domains. Beside well-known implementations for recommender systems [31] and self-driving cars [32], machine learning is widely spread in other fields of research such as geology [33], medicine [34] and physics [35]. For all of these applications the basic idea is to train a mapping with ground truth information. This ground truth is usually given by a set of training data which contains information about input variables and their corresponding output. Here, the output is distinguished between *regression* and *classification* applications. In particular, regression approximates a function which returns real numbers while classification identifies a specific group membership. However, depending on the particular type of training data, there are mainly three learning methods to be distinguished [36]:

- *Unsupervised learning* algorithms are trained to find hidden structures from input data without a predefined output. For this, output patterns are extracted during training and recognized for similar inputs during run-time, e.g., clustering.
- *Reinforcement learning* assigns a reward value to a sequence of input samples. The reward value can be understood as a simplified output, e.g., good-bad.
- *Supervised learning* techniques continuously generate discrete output estimates from the input variables and are also in the core of the presented machine learning approach.

Machine learning is further divided into lazy and eager learning algorithms [37]. Here, *lazy learning* is the comprehensive term for approaches where data processing is delayed until a query is made to the system. Once received, an answer is generated by a combination of the training data. The answer and intermediate results are discarded after this process. More precisely, lazy learner techniques fit real-time data to the training data, e.g., k-nearest neighbors, locally weighted regression or spline interpolation. This allows generating an answer for unknown situations from familiar ones. In the context of this thesis a novel interpolation scheme from fluid dynamics, the so called Dynamic Mode Decomposition (DMD) [38], is utilized. For example, this method is applied to generate a highly accurate stability model of a humanoid robot's walking gait (see Section 5.2). In contrast to that, *eager learning* algorithms generate an abstract model during a training phase. For example, Artificial Neural Networks (ANNs) learn to approximate the input/output function and discard the training data afterwards. Future requests are answered by generating a reply from the generated model.

Consequently, lazy learning requires less computational power than training an eager algorithm but usually have greater storage requirements. Furthermore, the response time of a lazy learner depends on the amount of training data while eager learning utilizes a compact model which usually replies in constant time. This thesis makes use of both eager and lazy algorithms to evaluate pros and cons of the presented BSPM approach in real world applications. Here, learning an accurate model usually involves the preprocessing of training data.

2.2 Preprocessing

The efficiency of machine learning algorithms primarily depends on the quality of the training data where a sufficient amount of heterogeneous training samples for the particular task need to be given. Here, sensor selection and dimensionality reduction techniques help to single out the most important features. To apply these techniques it is important that the data is in a useful scale and a unified format. This is usually achieved by performing two subsequent steps: *normalization* and *discretization*.

2.2.1 Normalization and Discretization

At present, robotic systems can acquire a rich set of sensor readings. These include joint angles, force-torque values or motor currents which rely on diverse measurement units. Depending on its particular position, the same kind of sensor can further have varying measuring ranges. For example, a humanoid robot's knee joint usually has a range of 90° while the shoulder has about 270°. This is solved by means of normalization [39] where the most popular methods are: *min-max* and *z-score* scaling. More precisely, min-max normalization linearly transforms a value x to the scaled value $\hat{x}$, which is in a range of $[0, 1]$, by

$$\hat{x} = \frac{x - x_{min}}{x_{max} - x_{min}}.$$

Here, x_{min} and x_{max} determine the possible minimum and maximum boundaries of the original data. In order to apply min-max normalization on robotic sensor streams, x_{min} and x_{max} of the particular sensor device need to be contained in the training data. Otherwise, larger or smaller real-time values would exceed the boundaries which requires to recalculate the normalization for all values and consequently to rerun the learning phase. One could also argue to utilize the particular hardware device limits to fix x_{min} and x_{max} only once. This requires a fair degree of knowledge about the robot's hardware. In contrast, the z-score scaling introduced by Larsen and Marx [40] normalize the data based on the mean $\bar{x}$ and standard deviation σ

$$\hat{x} = \frac{x - \bar{x}}{\sigma}.$$

Here, the particular sensor limits are not relevant to normalize the data. The authors also prove that data which follows any Gaussian distribution is transformed into a standard normal one.

Unfortunately, a wide variety of sensors return measurements in a noisy continuous space. For example, common force sensors return real numbers and are still prone to error. Therefore, they measure varying non-zero readings even if in rest. In turn, such slightly different values make a probability analysis less efficient. This problem is addressed by several quantization techniques which discretize the continuous space.

A popular method is the so called *binning* [41]. Here, the continuous space is portioned to a discrete one with a fixed number of uniform states. However, a further problem is that different sensors have distinct and even irregular frame rates. This is particularly critical for time dependent machine learning approaches which are trained with time-discrete sets of data. Hence, training and runtime data need to be acquired with identical frame rates or need to be synched by means of timestamps and interpolation schemes.

Finally, the preprocessed training data is analyzed for feature detection purposes. More precisely, information-theoretic measures or dimensionality reduction methods are applicable to select the important parts of the training data.

2.2.2 Information-Theoretic Sensor Selection

The use of information theory in sensor management was first proposed by Hintz [42] and found several applications in robotics.

In particular, Manyika and Durrant-Whyte [43] utilize information-theoretic measures to implement a data fusion and sensor management framework. Here, multiple sonar sensors of a mobile robot are analyzed and selected depending on their actual attention level. As a result the framework increases the overall navigation capabilities of the robot.

Moreover, Liu, Reich, and Zhao [44] present an information-driven vehicle tracking approach. They utilize a mobile platform which is equipped with an acoustic sensor network. This allows estimating the distance and direction to a given target. Here, mutual information is used to extract collaboration between the different sensors and

the target position. According to the authors, their approach is more energy efficient since less relevant sensors can be switched off temporarily.

Similar to this, Dame and Marchand [45] apply mutual information to a *visual servoing* task. More precisely, they control a six degree of freedom robotic arm from image data only. In contrast to other techniques, mutual information is insensitive to changes of lighting. This makes it suitable for detecting similar visual features which are used to align images under varying environmental conditions. As shown by the authors, the accuracy of the visual servoing approach benefits from the usage of mutual information and is also robust against large illumination variations.

Another possible technique to detect important correlations is to apply dimensionality reduction methods. In contrast to information theory, dimensionality reduction methods do not rely on the selection of raw data. Instead, the complete data is preprocessed to a compact representation which maintains the more important parts stronger than the less relevant.

2.2.3 Dimensionality Reduction

Dimensionality reduction involves a mapping from high-dimensional input points to low-dimensional manifolds and vice versa. This allows increasing the efficiency of the corresponding lazy or eager learning technique.

For example, Ciocarlie, Goldfeder, and Allen [46] applies dimensionality reduction for grasps performed by a robotic hand. Here, the grasp is controlled by a highly reduced configuration space which is built upon Principal Component Analysis (PCA). More precisely, PCA extracts the latent structure of training data based on eigenanalysis of covariances and applies a linear mapping between the different dimensionalities. The generalizability of the generated low-dimensional configuration space is shown by the implementation of a grasp planner which is applied to significantly different hands.

Similar to this, Artemiadis and Kyriakopoulos [47] utilize PCA to generate a low-dimensional embedding from high-dimensional electromyographic signals. These signals are measured at the upper limb of a human and therefore represent muscle synergies and motion primitives. A three-dimensional projection to Cartesian coordinates is then utilized for the continuous control of a robotic arm. The overall approach is implemented in the context of a human-robot teleoperation interface.

In contrast, Bitzer, Howard, and Vijayakumar [48] make use of a nonlinear extension to PCA based on Gaussian processes. Here, inherent structures are extracted from the training data and used as compact state representation. These states are then utilized to learn policies which enable a humanoid robot to solve bi-manual reaching tasks.

Following Hinton and Salakhutdinov [49], also ANNs are a promising technique for reducing the dimensionality of data efficiently. In particular, an *autoencoder* network contains multiple layers with a small central layer. These autoencoders are trained to reconstruct high-dimensional inputs where the central layer is used as low-dimensional representation. According to Hinton and Salakhutdinov and similar to previous findings, the accuracy of autoencoders increase with their depth and can even outperform

the accuracy of PCA.

However, Ben Amor [50] compared a number of dimensionality reduction techniques used to extract relevant manifolds from motion data. As proven by the author, PCA does not achieve the highest accuracies but generalizes well for various motions and significantly different dimensionality while requiring less computational effort than most other methods.

2.3 Force Estimation in Robotics

Robot platforms, which assist or replace humans in physically demanding tasks, need precise force sensing capabilities. As noted by Haddadin, De Luca, and Albu-Schäffer [51], safe and efficient interaction further requires to detect and handle collisions for the entire robot platform. Instead of utilizing a large number of special purpose sensors, this can be achieved by generating force estimates from the robot's integrated proprioceptors. For example, industrial robot manipulators such as the UR5 utilize *analytical* machine learning approaches to estimate forces acting at the end-effector. Such analytical approaches are usually built upon an accurate mathematical representation of the system dynamics.

For example, Stolt et al. [52] utilize the robot's control loop architecture during performing assembly tasks. In particular, the robot's joint position control errors are utilized for force estimation at its gripper. As stated by the authors, the accuracy of these estimates depends on a task-specific modification of the control loop mechanism. This requires an adequate configuration of an empirically determined scaling parameter.

Furthermore, unknown environmental constraints, e.g., friction, decrease the accuracy of the force estimates when the robot is not moving. Erhart, Sieber, and Hirche [53] present a related impedance-based control architecture for a cooperative robot-robot transportation task. Here, both robots and the rigidly grasped object are considered as a closed kinematic chain where position/orientation of the end-effectors and the inertia/mass of the object need to be known. Given an accurate configuration, undesired forces within the closed kinematic chain can be limited by real-time adaptation of the end-effectors.

Wahrburg et al. [54] estimate contact forces for a redundant robotic manipulator. The redundant joints of the robot allow reaching the same end-effector position with different joint-configurations which influences the quality of the force estimation. Here, the momentum is analyzed from the robot's joint angles/speeds, motor torques and the manipulator dynamics. This enables the robot to find an optimal joint configuration for a given end-effector position.

In general, the quality of such analytical approaches depends on an accurate configuration and precise knowledge about system dynamics. The required amount of expert-knowledge is even intensified for more complex nonlinear systems which is a major limitation of analytical approaches. In contrast, humans do not rely on such precise mathematical models and instead learn to approximate highly nonlinear system dynamics from prior experience. Similar to this, *data-driven* machine learning

approaches involve a training phase where the task is demonstrated to the robot. The acquired training data is then utilized to learn an abstract model which approximates the system dynamics.

For instance, Colomé et al. [55] learn an inverse model of the robot dynamics where only the inertia matrix of the robot needs to be given. Here, locally weighted projection regression [56] is utilized to map desired joint positions/velocities to the required motor torque commands for a predefined trajectory. Comparing the predicted torque values with the actually applied ones allows extracting the amount and direction of external forces. This enables a robotic arm to measure the weight attached to its end-effector. However, only a single trajectory is learned while the generalizability to other end-effector positions is not proven by the authors.

He et al. [57] utilize a learning procedure for physical human-robot interaction in rehabilitation. Here, neural networks are trained to approximate the unknown dynamics of the robot and to adapt the behavior with regard to the forces applied by the human interaction partner. However, the approach is verified inside a simulated environment while its validity in real world applications still needs to be proven.

Schröder-Schetelig, Manoonpong, and Wörgötter [58] present a dynamic walking application on a biped robot platform. The motor commands are used as input to the forward model which predicts the intrinsic acceleration during walking. The robot's actual acceleration is then compared with the predictions made and used to control the upper body component. This allows compensating external forces which originate from altered terrain and consequently stabilizes the walking gait.

To summarize, data-driven approaches are capable to learn robot dynamics without the need of expert-knowledge about the particular robot but are usually restricted to manual sensor selection, a single trajectory and are demonstrated for only one specific robot platform. In contrast, this thesis presents a data-driven approach which is applicable to a wide range of robot platforms, provided sufficient access to proprioceptive sensors is given. The approach has been applied to different behaviors, e.g., walking, tightening, lifting, and is also flexible to environmental conditions such as different lifting positions or force sources. Here, the most relevant proprioceptors are selected automatically which increases the efficiency of the utilized model learning techniques, e.g., neural networks.

2.4 Artificial Neural Networks

The presented machine learning approach is based on the fundamental idea of giving a robotic system the ability to learn from previous demonstrations. A common technique to learn from a set of training data in a supervised fashion are ANNs. This is due to several advantages over lazy and other eager learning approaches:

- They are able to approximate the underlying linear or nonlinear function contained in the training data.

- They can easily handle a huge amount of training data and can be scaled to any number of input variables.
- Once training is finished, the network's computational demand is independent from the particular input and respond in constant time.

Hence, ANNs are a promising technique for the implementation of the presented BSPM approach. In the following, a historical perspective to the field of ANNs is given. Next, the general applicability of ANNs in various fields of research is examined. Finally, several state-of-the-art deep neural architectures up to the extraction of indefinite temporal delays are discussed.

2.4.1 Historical Perspective

Such ANNs found their first practical application by the research of Rosenblatt and its definition of *perceptrons* in 1957 [59][60]. The perceptron was actually planned to be implemented directly in a custom-built neurocomputer, the so called *Mark 1 perceptron* [61]. This machine utilizes 400 photocells (input neurons) which are randomly connected to output neurons. The generated estimates are used for image recognition, e.g., to classify numbers. For evaluation purposes it was, in parallel, implemented as software which therefore was finished even earlier then the hardware version. This perceptron contains exactly one layer of adaptable weights and is also referred to as Single-Layer Perceptron (SLP). Furthermore, SLPs contain the basic building blocks: neurons, connections and weights and therefore are still the base of nowadays ANN research.

However, after a period of growth the research was abruptly slowed down after in 1969 Minsky and Papert published a precise mathematical analysis about SLPs [62]. They proved that a SLP is not capable of representing nonlinear functions as the XOR problem. Hence, they forecasted that the entire field of research is a dead end. In 1986 Rumelhart, Hinton, and Williams [63] disproved these negative evaluations. For this, they proposed Multi-Layer Perceptrons (MLPs) which are capable of solving also nonlinearly separable problems. The main advantage of MLPs is that they can have an unlimited number of layers which are hidden from the outside. With an increasing number of such hidden layers the network structure gets deeper and more complex. The challenges and improvements arising from organizing and learning such structures are referred to as *deep learning* [64]. A detailed explanation of SLPs, MLPs and deep neural network architectures is given in Section 3.2. Starting from the functional discoveries of MLPs, the field of ANNs are almost grown and found applications in many fields of research.

2.4.2 General Applicability

A common application of ANNs is to forecast a sequence of successively measured points in time. Such kind of time series data can be found in different fields of application such as mathematical finance, econometrics and earthquake prediction.

For example, Zhang and Hu [65] present an ANN based approach to forecast the exchange rate of the British pound and the US dollar. Based on a huge set of financial data, they examined the effects of the network configuration and the amount of training samples on the prediction accuracy. As a result, they found that the prediction accuracy benefits from an increase of input variables and a large number of training samples.

Osman, Awad, and Mahmoud [66] make use of ANNs for short-term load forecasting for the Egyptian Unified System. Such predictions are essential for electric power system planning and are strongly depending on the actual weather, temperature and seasonal variations. In contrast to previous approaches, the authors utilize a correlation based selection scheme of actual weather data. By utilizing this subset of input variables for the ANN results in accurate predictions and makes the approach robust against rapid temperature changes.

Panakkat and Adeli [67] are using seismicity indicators to forecast location, time of occurrence and the magnitude of earthquakes. A large seismic data of Southern California is preprocessed by (1) dividing into subregions or (2) splitting it into temporal subsets. For (1) seismicity indicators are derived from the magnitude of the largest earthquake while (2) make use of the latitude and longitude of the epicenter. These indicators are used as the input variables of an ANN where (2) is found to produce more accurate results than (1). According to the authors, the final indicators produce the best results in regions with intense seismic activity since also more seismic data is available.

Even if located in very different fields of research, all of these applications came to very similar conclusions where the crucial factors for accurate and robust forecasting of ANNs rely on

- the availability and selection of relevant input variables
- and the amount and heterogeneity of training samples.

Similar to the introduced applications, robotic platforms are also providing time series sensor readings such as joint angles, motor currents or force-torque values. Therefore, data acquisition and input selection schemes are also at the core of the proposed machine learning approach.

Besides applications related to time series prediction, ANNs are also widely applied to control robotic behavior [68]. For example, Velagic, Osmic, and Lacevic [69] present a motion controller for mobile robots. Here, the goal is to follow a given trajectory while feedback is obtained from the robot's odometry. The position errors are used as input variables of a MLP and are mapped to the velocity which fixes the particular position errors. However, the results are only demonstrated inside a simulation while applicability and validity still has to be proven in real world experience.

Tsai, Huang, and Lin [70] apply ANNs to control the balancing of a two-wheeled robotic platform. They decompose the overall vehicle system into two adaptive MLP controllers where (1) regulates the yaw motion and (2) balances roll and pitch. (1) makes use of potentiometer measurements while (2) utilize a gyroscope and a tilt sensor as input data. The overall system allows maintaining the human body on the

footplate without falling and is even taken into account two-dimensional friction. Due to an inconvenient reaction time, the proposed approach is limited to low velocities.

Another application presented by Kumar et al. [71] makes use of an ANN for the adaptive control of robot manipulators. For this, they determine the robot's dynamics by means of force, position and redundant joint subspaces. The given dynamics is then used to preset the parameters of a MLP where the errors of force, position and velocity are used as input. In their experiments they apply the presented approach on a simulated two link robotic manipulator with known and unknown dynamics. The authors show that by utilizing knowledge about the robot's dynamics can increase the learning curve but requires quite a lot expert knowledge and intense user effort.

Similar to these applications, the proposed machine learning approach makes use of sensor readings to generate an adequate reaction. Concluding from the drawbacks of the presented applications this thesis aims on the implementation of self-adaptable behaviors in

- real world applications,
- with a constantly low reaction time
- and without the need of expert knowledge.

In order to equip robots with accurate interaction capabilities they need to perceive their environment [15]. Hence, a common problem in computer vision is the detection of patterns, e.g., detecting faces or finding objects in the environment.

The first rotation invariant face detection approach was introduced 1998 by Rowley et al. [72]. By splitting the image into small sized windows a first MLP is utilized to determine the orientation angle of faces. The derotated image window is preprocessed by means of lighting and contrast before send to another framework of MLPs. These return a value in range between plus one (face) and minus one (no face). However, for rotations within the image plan the accuracy of the proposed approach is about 80% which was a major advance at that time.

Viola and Jones [73] achieve similar accuracies but further allow to detect arbitrary objects more rapidly. For this, an image is decomposed into sub-images and regions of interest. The smaller images are used for calculating a set of features in constant time. The most important features are selected as input for a cascade structure of ANN classifiers. Here, promising regions receive more attention which increases the performance of the presented approach.

To sum up, the introduced applications highlight the importance of feature selection. In contrast to the introduced pattern detection/recognition in computer vision, this thesis aims on the relation between motion and haptic stimuli. Since cognitive skills involve the visual as the haptic sense, it is assumed that deep learning architectures are also beneficial for the presented machine learning approach. Therefore, several state-of-the-art deep ANNs are introduced in the following.

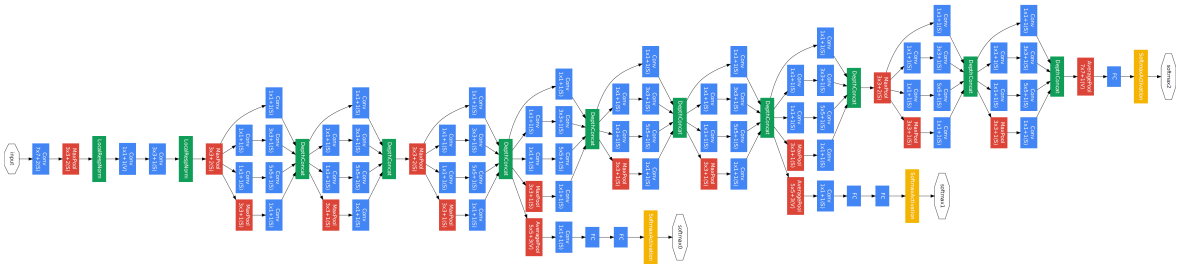


Fig. 2.1: An overview of the GoogLeNet architecture with nine inception modules, twenty-two sequential layers performs about one hundred parallel operations. The utilized inception modules allow implementing parallel units which have the ability to perform multiple operations at once. adopted from [75, p. 6]

2.4.3 Deep Neural Network Architectures

In order to achieve a better comparability between the increasing number of approaches Deng et al. introduced *ImageNet* [74], which is a hierarchical database with meanwhile more than 14 million static images. This database contains a wide variety of objects, e.g., animals, plants, devices, tools, structures, vehicles and persons making it applicable for most fields of research. ImageNet therefore is used for evaluation purposes of state-of-the-art classification architectures.

Namely, Krizhevsky, Sutskever, and Hinton [76] utilize it to evaluate *AlexNet*, a deep Convolutional Neural Network (CNN) architecture. Such CNNs have outstanding classification capabilities [77] but are not specifically designed to handle temporal relations. Hence, they are usually applied to static data. Therefore, AlexNet is trained to classify 1000 categories from the 1.2 million high-resolution images contained in ImageNet. Its structure contain an overall of 60 million parameters and 650 thousand neurons distributed among eight weight layers and requires an overall of 6 GB memory. After six days of training, the resulting network is capable to classify objects with a *top-5 error* of 15.4%. Here, the top-5 error is the rate at which the correct object is not contained in the five most probable results. To give an idea about the complexity of this problem, humans in general achieve accuracies within between $5 - 10\%$.

However, according to the authors the depth of AlexNet is vital for achieving high classification accuracies and is limited by two factors: the amount of memory and the time spend for training. The importance of depth is further proven by the studies of Zeiler and Fergus [78] who also introduced the *ZF Net* architecture. ZF Net is a slightly adapted version of AlexNet and achieves a top-5 error rate of 11.2% for the ImageNet classification task. Inspired by this discoveries, very deep convolutional networks and the *VGG Net* architecture was introduced by Simonyan and Zisserman [79]. The VGG Net is constructed of 19 weight layers and has a memory demand of 24 GB. The resulting top-5 error rate after about two weeks of training is 7.3%. Therefore, its accuracy is comparable to the cognitive capabilities of humans. The authors further investigated the beneficial effects of depth on the accuracy of visual representations

Tab. 2.1: A summary of the reviewed methodologies. Here, the error rate is defined by the top-5 error rate when applied to high-resolution images contained in the ImageNet [74] data set. Following Krizhevsky, Sutskever, and Hinton [76], depth is the crucial factor for the classification capabilities of a network which therefore is restricted by means of memory and computational power spent for training.

	year	layers	error rate	hardware	training
AlexNet	2012	8	15.4	2×GPU	$\sim 6\,\mathrm{d}$
ZF Net	2013	8	11.2	1×GPU	$\sim 12\,\mathrm{d}$
VGG Net	2014	19	7.3	4×GPU	$\sim 14\,\mathrm{d} - 21\,\mathrm{d}$
GoogLeNet	2015	22	6.7	CPU cluster[1]	—
MicrosoftResNet	2015	152	**3.6**	8×GPU	$\sim 14\,\mathrm{d} - 21\,\mathrm{d}$

and confirmed the assumptions of Krizhevsky, Sutskever, and Hinton [76].

In 2015 Szegedy et al. [75] introduced and applied *inception module* to build the *GoogLeNet* architecture. These modules contain a number of parallel units. Hence, in contrast to the naive sequential processing of information, multiple operations can be performed at once. GoogLeNet utilizes nine of these modules with a depth of 22 sequential and about 100 parallel units and is illustrated in Figure 2.1. The authors state that the resulting model requires a notable amount of memory and power usage. Therefore, training was performed on the *DistBelief* [80] CPU cluster achieving a top-5 error rate of 6.7%.

Later in 2015, He et al. achieved a further breakthrough by introducing *Microsoft's Residual Network* also termed *MicrosoftResNet* [81]. Their fundamental contribution is given by the introduction of so called *residual blocks*. Such blocks allow information to skip a series of layers. Hence, the input received by lower layers is also available to a node in a higher layer. In contrast to previous approaches, this architecture is easier to train which allows increasing the number of layers incredibly. More precisely, MicrosoftResNet is constructed by 152 layers but required a comparatively little training effort of two to three weeks on a cluster of eight graphical processing units. This has a huge impact on the classification capabilities which for the top-5 error rate achieve an accuracy of 3.6%. Consequently, the achieved image recognition capabilities of this artificial intelligence approach are beyond human cognition.

Table 2.1 summarizes the reviewed methodologies. Given those developments, one can clearly see the ongoing progress in the field which further highlights the importance of depth. This is also mathematically proven by recent research of Eldan and Shamir [82] who show that depth the key for such ANNs. Due to the restrictions of CNNs, the introduced approaches do not take into account image sequences which may provide beneficial temporal relations. Such temporal relations are the fundamental idea behind Long Short-Term Memorys (LSTMs) [83] which are discussed in the following.

[1] The DistBelief software framework [80] allows utilizing large-scale computing clusters.

2.4.4 Temporal Delays

The ability to remember past information is the core idea behind Recurrent Neural Networks (RNNs). These networks contain recurrent connections which provide time delayed information about the network's inputs, outputs or neuron states. This allows the network to combine the actual inputs with its past and is substantial for temporal related input data, e.g., image sequences. One of the most commonly used recurrent network architecture is the so called LSTM. In contrast to spreading neurons over a huge amount of layers, LSTM consists of more complex *blocks* which are usually distributed among one or two layers. A major advance of these blocks over classical neurons is that they are able to maintain information indefinitely. Hence, achieving similar capabilities with a CNN would require an infinite number of layers.

As found by Srivastava, Greff, and Schmidhuber [84], residual blocks are a simplified implementation of LSTM blocks. Hence, findings related to CNNs can be beneficial for the development of LSTM applications. Beside its adaptation for image classification, LSTMs are usually applied to temporal structured data such as text, audio and video.

For example, recent research of Venugopalan et al. [85] combines CNN and LSTM techniques to extract video descriptions. Here, static image information is extracted from a sequence of frames by utilizing a CNN. The resulting outputs are used as sequential input for an LSTM which therefore generates a sequence of words. This sequence is trained to describe the actual scene of the video sequence, e.g., 'A man is cutting a bottle'. As noted by the authors, this is the first sequence to sequence model for generating video descriptions.

Furthermore, Eck and Schmidhuber [86] apply LSTM to find temporal structures in blues music. Here, the main problem is that the style of the music is given by the global structure of the overall signal. The network therefore needs to keep track of temporal distant events which is also the core motivation for using LSTM. Given several blues music training sets the network was trained to predict the probability for a note. By applying the learned structure, the LSTM is able to constantly generate blues music from its own previous compositions. According to the authors, the generated music is following the structure of blues and does not drift away.

Moreover, Sundermeyer, Schlüter, and Ney [87] utilize LSTM for modeling the French and English language. The proposed network is able to take into account the given context to estimate the probability for each word and also to predict the next word. Here, a CNN would only take into account a fixed context length while LSTMs are able to handle all previous words. Such capabilities are important to understand the meaning of words and phrases depending on their particular sequence. Therefore, related fields of application include handwritten text recognition, speech recognition and machine translation.

To summarize, LSTM is capable of remembering information indefinitely. This allows to analyze a complete sequence of time series for its

- underlying core structure and
- correlations between time delayed events.

Learned from sequences of training data, these networks can be used for regression and classification purposes and already found diverse applications in robotics.

In particular, Förster, Graves, and Schmidhuber [88] utilize LSTM for robot localization. Here, the training data is derived from a set of two-dimensional laser range distance sensors and roughly generated labels of the environment. The obtained data streams are used as network input while a specific area label is used as output. Despite the lack of real world implementations, it enables a simulated robot to accurately localize itself.

The work of How et al. [89] make use of LSTM to recognize behaviors performed by a real world humanoid robot. The input to the network is derived from sequences of joint angle data while the output applies a probability distribution to behavior-specific classes. A major challenge of this task is to correctly classify behaviors which are constructed of similar parts. For instance, behaviors with similar endings require a model which is capable to bridge long time lags. As shown by the authors, this was accurately achieved by the possibly infinite time delay of LSTM.

Besides that, Otte et al. [90] utilize an LSTM network to human trajectory prediction. Such trajectories are important for the deployment of socially-aware robots [91]. More precisely, robots need to be able to forecast the position of pedestrians to adapt its own path and behavior. Especially in crowed spaces, this is essential for collision avoidance and human-robot interaction tasks. For this, one LSTM is trained for each person in a scene. As a result, the corresponding LSTM fits the particular constraints of the person, e.g., velocity, acceleration and gait style and predicts the future position for the particular person. By connecting the different LSTM networks the overall model is able to imply trajectory relations, such as collision avoidance, into the corresponding predictions.

In contrast to recent work, this thesis implements LSTM for proprioception tasks. For example, this enables a robotic arm to precisely classify weight without the need of a special purpose sensor (see Section 6.2).

2.5 Conclusion

In this thesis, biological inspired concepts such as proprioception and neural networks are combined with probability and information theory. Similar to the human sense of self a training phase is required to acquire behavior-specific ground truth data from the robot's sensor readings. After experiencing a behavior, the most relevant information is used as training data for learning supervised models of proprioception.

Here, the learning efficiency depends on the quality and size of training examples but also on an appropriate preprocessing. In particular, this thesis applies different information-theoretic measures for sensor selection and feature extraction purposes. Furthermore, dimensionality reduction can be applied to generate a low-dimensional embedding from high-dimensional training data. The preprocessed training data is then applied to a lazy or eager learning technique which represents the corresponding BSPM. As a result, the presented approach equips robots with a sense of self.

The goal of the presented approach is to achieve similar or even better force estimation capabilities as experienced workers where the following goals and restrictions highlight the difference to previous research:

- The approach is not restricted to a particular platform and can be applied to arbitrary robotic behaviors.
- A minimum of expert knowledge about the particular hardware and the executed behavior is required.
- To ensure interactivity and safety issues a reaction need to be triggered within a constantly low reaction time.
- The approach is practicable for real world applications without the need of an additional simulated environment.
- The learning efficiency is increased by making use of correlations within the robot's sensory-motor integration.
- The approach is built upon the robot's integrated proprioceptors, utilize a minimum of special purpose hardware and requires no extraordinary computational power to make a low-cost implementation possible.
- Only a small number of training examples is required which shortens the training phase and consequently reduces the wear and tear of the robotic hardware as well as the effort spend by the user.

In general, these requirements ensure the applicability of the presented approach. The following chapter examines the mathematical foundations of the utilized information-theoretic measures and neural network architectures in more detail.

3 Mathematical Foundations

This chapter introduces the mathematical fundamentals for the presented BSPM approach. First, information measures, which can be applied to detect simultaneous and delayed dependencies between a robot's proprioception and the behavior-specific context, are introduced. Next, the usage of Artificial Neural Networks (ANNs) for function approximation is described and successively extended to state-of-the-art deep learning techniques. Finally, a conclusion about these techniques and their application for BSPMs is given.

3.1 Information Measure

The field of information theory was proposed by *Claude Elwood Shannon* [92] with the primarily goal to determine the amount of information which can be transmitted by a particular communication channel. Meanwhile, this powerful theory is widely spread in other research areas, e.g., anomaly detection, pattern recognition and model selection. Therefore, it is generally applied to detect correlations between arbitrary processes.

In the following, the most relevant basics of the information theory are introduced (Section 3.1.1). Furthermore, two techniques, which allow to measure shared information (Section 3.1.2) and information transfer (Section 3.1.3), are presented.

3.1.1 Shannon Entropy

A straightforward way to acquire knowledge about a discrete random variable $\mathcal{X}$ (or process) is to measure its average information content. For this, the probability function $p(\mathcal{X})$ for each possible state $\{x_1, \ldots, x_n\} \in \mathcal{X}$ needs to be given. In practice, this is usually done by sampling $\mathcal{X}$ until a statistically significant amount is reached. For each possible state of $\mathcal{X}$ the information content is determined as

$$I(x) = -\log_a p(x), \tag{3.1}$$

where $p(x)$ is the probability of the particular state and a describes the unit measuring the information content. In particular, for $a = 2$ the information content is measured in *bits* or *shannon* and is usually used in the field of information theory. The resulting information content $I(x)$ specifies the minimal number of bits needed to represent the particular state $x \in \mathcal{X}$. Figure 3.1 left illustrates the relation between probability and information content of a state. Rare states have a low probability and therefore high information content while more frequent states have a higher probability and therefore

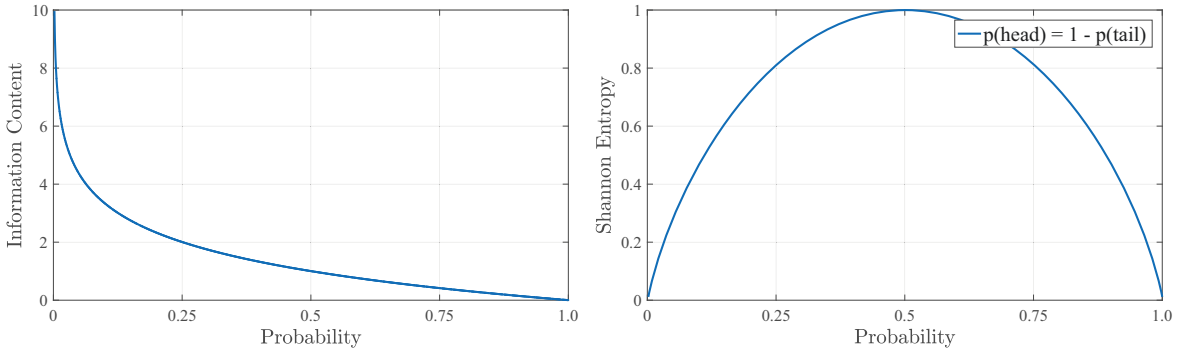


Fig. 3.1: The relations between probability, information content and Shannon entropy. Left: The relation between probability and information content of a state. For decreasing probabilities the information content of a state converges toward infinite. A state contains no information if it has a probability of one. Right: The relation between the probabilities of tossing a coin and the Shannon entropy. The Shannon entropy decreases the stronger the probabilities differ what therefore simplifies the prediction of future states.

lower information content.

Depending on the particular probability, the weighted summation of the information content for all states of $\mathcal{X}$ is given by the Shannon entropy:

$$H(\mathcal{X}) = \sum_{i=1}^{n} p(x_i) I(x_i) = -\sum_{i=1}^{n} p(x_i) \log_a p(x_i). \tag{3.2}$$

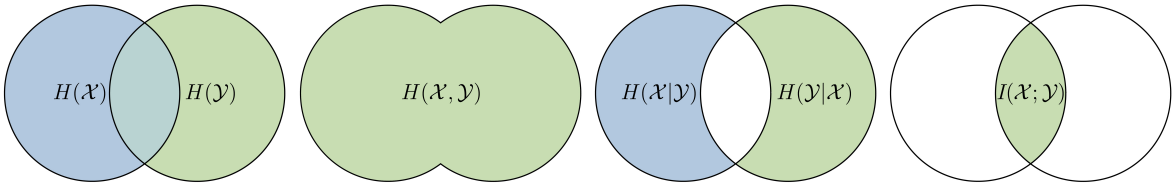
Hence, the Shannon entropy describes the average number of bits needed to encode a state of $\mathcal{X}$. In the following, this is exemplified by tossing a coin where the state of $\mathcal{X}$ is either heads or tails.

For a fair coin the probability for head (p_{head}) or tail (p_{tail}) is both $\frac{1}{2}$. Utilizing Equation 3.2, the corresponding Shannon entropy is

$$\begin{aligned} H_{fair} &= -[(p_{head} \log_2 p_{head}) + (p_{tail} \log_2 p_{tail})] \\ &= -[(\tfrac{1}{2} \log_2 \tfrac{1}{2}) + (\tfrac{1}{2} \log_2 \tfrac{1}{2})] \\ &= -[(-\tfrac{1}{2}) + (-\tfrac{1}{2})] \\ &= 1 \,\text{bit}. \end{aligned} \tag{3.3}$$

Thus, the Shannon entropy of tossing a fair coin is one bit and therefore at least one bit is required for the encoding.

A further advantage of the Shannon entropy is that it can be interpreted as the magnitude of uncertainty about the future state of the process. This is illustrated by another coin experiment with 'unfair' probabilities $q_{head} = \frac{3}{4}$ and $q_{tail} = \frac{1}{4}$. Obviously, the future state of the unfair coin should be easier to predict than the fair one. The



(a) Shannon Entropy (b) Joint Entropy (c) Conditional Entropy (d) Mutual Information

Fig. 3.2: The relations between two not independent processes $\mathcal{X}$ and $\mathcal{Y}$. The joint entropy is the union of $H(\mathcal{X})$ and $H(\mathcal{Y})$ while the conditional entropy is the independent part of the particular process. In contrast to that, mutual information is the amount of information both processes share and therefore is interpreted as the reduction of uncertainty about the processes when the other is known.

Shannon entropy for the unfair coin is

$$\begin{aligned} H_{unfair} &= -[(q_{head}\log_2 q_{head}) + (q_{tail}\log_2 q_{tail})] \\ &= -[(\tfrac{3}{4}\log_2 \tfrac{3}{4}) + (\tfrac{1}{4}\log_2 \tfrac{1}{4})] \\ &= -[(-0.3112) + (-\tfrac{1}{2})] \\ &= 0.8112\,\text{bit}. \end{aligned} \tag{3.4}$$

The unfair coin carries less information than the fair one and therefore is easier to predict. Figure 3.1 right illustrates the relationship between the probability for $p(head) = 1 - p(tail)$ and the Shannon entropy. As can be seen, the Shannon entropy decreases when the future state of the coin gets easier to predict. Hence, Shannon entropy can be used as a measurement of predictability of a process.

However, Shannon entropy only takes into account the state probabilities of one independent process while the actual state of the process and other processes are omitted. In order to determine the predictability of a process more accurately, the following two extensions are substantial and need to be taken into account:

1. influences from simultaneous processes
2. time delayed dependencies

(1) is investigated by measuring shared information as explained in the following section while (2) is taken into account by utilizing information transfer described in Section 3.1.3.

3.1.2 Mutual Information

In the following, dependencies between simultaneous processes are uncovered. The main idea is to identify processes which are influencing other processes and therefore are not independent. A process $\mathcal{X}$ is called independent if the knowledge about another process $\mathcal{Y}$ does not reduce the uncertainty about $\mathcal{X}$, where $p(\mathcal{X})$ and $p(\mathcal{Y})$ are the corresponding probability functions. The amount of information remaining for a process $\mathcal{X}$ when knowing a process $\mathcal{Y}$ is quantified by the *conditional entropy*

Tab. 3.1: 20 repetitions of tossing a coin. $\mathcal{X}$ is the result of the coin (heads are zeroes and tails are ones) while process $\mathcal{Y}$ is a trigger to manipulate the flip of the coin which always result in tails for process $\mathcal{X}$.

$\mathcal{X}$	0	1	**1**	1	0	1	0	**1**	0	1	0	1	**1**	0	0	0	1	0	**1**	**1**
$\mathcal{Y}$	0	0	**1**	0	0	0	0	**1**	0	0	0	0	**1**	0	0	0	0	0	**1**	**1**

$$H(\mathcal{X}|\mathcal{Y}) = H(\mathcal{X},\mathcal{Y}) - H(\mathcal{Y}), \tag{3.5}$$

where $H(\mathcal{X},\mathcal{Y})$ defines the *joint entropy* of a set of processes

$$H(\mathcal{X},\mathcal{Y}) = H(\mathcal{Y},\mathcal{X}) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log_a p(x,y). \tag{3.6}$$

The relations between Shannon entropy, joint entropy and conditional entropy are illustrated in Figure 3.2(a) to 3.2(c). As can be seen, the joint entropy $H(\mathcal{X},\mathcal{Y})$ is the union of $H(\mathcal{X})$ and $H(\mathcal{Y})$ while conditional entropy describes the independent part. Consequently, process $\mathcal{X}$ completely depends on $\mathcal{Y}$ when $H(\mathcal{X}|\mathcal{Y}) = 0$ and is independent from $\mathcal{Y}$ when $H(\mathcal{X}|\mathcal{Y}) = H(\mathcal{X})$. The reduction of uncertainty about a process $\mathcal{X}$ when $\mathcal{Y}$ is known is described by the *mutual information*

$$\begin{aligned} I(\mathcal{X};\mathcal{Y}) &= H(\mathcal{X}) - H(\mathcal{X}|\mathcal{Y}) \\ &= H(\mathcal{X}) + H(\mathcal{Y}) - H(\mathcal{X},\mathcal{Y}) \\ &= \sum_{x\in\mathcal{X}} p(x)\log_a\frac{1}{p(x)} + \sum_{y\in\mathcal{Y}} p(y)\log_a\frac{1}{p(y)} + \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log_a p(x,y) \\ &= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log_a\frac{p(x,y)}{p(x)p(y)}. \end{aligned} \tag{3.7}$$

Thus, mutual information measures the amount of information produced by the premise that the processes $\mathcal{X}$ and $\mathcal{Y}$ are independent. Or to put it more simply, mutual information quantifies the amount of information both processes share (Figure 3.2(d)). As a result, the amount of mutual information increases with the Shannon entropy as well as with the dependency of the processes and will be zero if the processes are independent. However, a main drawback of mutual information is its symmetric property $I(\mathcal{X};\mathcal{Y}) = I(\mathcal{Y};\mathcal{X})$. To show this, a slightly adapted coin experiment is considered.

This time the result of the coin, represented as process $\mathcal{X}$, is manipulated by a trigger process $\mathcal{Y}$. Whenever the state $y \in \mathcal{Y}$ equals one, a trigger is activated which flips $x \in \mathcal{X}$ to one. As long as the trigger is not activated, the coin behaves like a fair one. Table 3.1 shows 20 repetitions of this experiment while the corresponding probability distributions are listed in Table 3.2. For the manipulated coin the probability for head ($\frac{9}{20}$) or tail ($\frac{11}{20}$) is still close to a fair one. In contrast to that, the mutual information

Tab. 3.2: The corresponding probabilities of the 20 coin tossing repetitions shown in Table 3.1.

x	y	$p(x,y)$	$p(x)$	$p(y)$
0	0	$\frac{9}{20}$	$\frac{9}{20}$	$\frac{15}{20}$
0	1	0	$\frac{9}{20}$	$\frac{5}{20}$
1	0	$\frac{6}{20}$	$\frac{11}{20}$	$\frac{15}{20}$
1	1	$\frac{5}{20}$	$\frac{11}{20}$	$\frac{5}{20}$

calculated by Equation 3.7 uncovers the influence of the manipulating process $\mathcal{Y}$

$$\begin{aligned} I(\mathcal{X};\mathcal{Y}) &= \left(\frac{9}{20}\cdot\log_2\frac{\frac{9}{20}}{\frac{9}{20}\cdot\frac{15}{20}}\right)+\left(\frac{6}{20}\cdot\log_2\frac{\frac{6}{20}}{\frac{11}{20}\cdot\frac{15}{20}}\right)+\left(\frac{5}{20}\cdot\log_2\frac{\frac{5}{20}}{\frac{11}{20}\cdot\frac{5}{20}}\right) \\ &= \left(\frac{9}{20}\cdot 0.415\right)+\left(\frac{6}{20}\cdot -0.4594\right)+\left(\frac{5}{20}\cdot 0.8625\right) \\ &= 0.2646\,\text{bit}. \end{aligned}$$

Hence, if the state of the trigger is known, the uncertainty about the state of the coin (its Shannon entropy) is reduced by 0.2646 bit. Broadly speaking, it is easier to determine the actual state of $\mathcal{X}$ when the actual state of $\mathcal{Y}$ is known. Since $I(\mathcal{X};\mathcal{Y}) = I(\mathcal{Y};\mathcal{X})$, also the uncertainty about the state of the trigger is reduced by the same amount if the state of the coin is known. This missing directional sense disables mutual information to identify either the coin is influenced by the trigger or vice versa and therefore cannot distinguish between causal dependencies and spurious correlations.

However, so far only the mutual information between two processes was taken into account. In order to detect correlations between an arbitrary number of simultaneous processes, the concepts of Multivariate Mutual Information (MMI) and Conditional Mutual Information (CMI) are introduced. As illustrated in Figure 3.3, a Venn diagram is utilized to illustrate the relations between these information-theoretic measures in a simplified manner. Here, the MMI, quantifies the amount of information shared between arbitrary processes. In the context of this thesis, it is sufficient to determine the dependencies between three processes

$$\begin{aligned} I(\mathcal{X};\mathcal{Y};\mathcal{Z}) &= I(\mathcal{X};\mathcal{Y}) - I(\mathcal{X};\mathcal{Y}|\mathcal{Z}) \\ &= I(\mathcal{X};\mathcal{Z}) + I(\mathcal{Y};\mathcal{Z}) - I(\mathcal{X},\mathcal{Y};\mathcal{Z}), \end{aligned} \tag{3.8}$$

where $I(\cdot;\cdot|\cdot)$ denotes the CMI. More precisely, the CMI quantifies the amount of information shared between the processes $\{\mathcal{X},\mathcal{Y}\}$ when $\mathcal{Z}$ is known

$$I(\mathcal{X};\mathcal{Y}|\mathcal{Z}) = H(\mathcal{X},\mathcal{Z}) + H(\mathcal{Y},\mathcal{Z}) - H(\mathcal{X},\mathcal{Y},\mathcal{Z}) - H(\mathcal{Z}). \tag{3.9}$$

Hence, the MMI $I(\mathcal{X};\mathcal{Y};\mathcal{Z})$ describes the information gained by taking into account $I(\mathcal{X},\mathcal{Y};\mathcal{Z})$ instead of a pairwise combination $I(\mathcal{X};\mathcal{Z}) + I(\mathcal{Y};\mathcal{Z})$.

As stated above, Figure 3.3 gives a simplified insight of potentially complex depen-

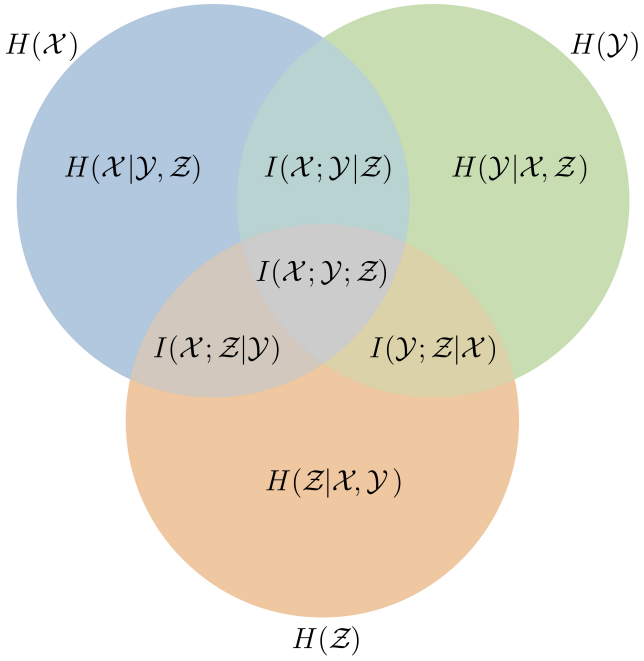


Fig. 3.3: A Venn diagram is utilized to explain the correlations between three partially dependent processes $\{\mathcal{X},\mathcal{Y},\mathcal{Z}\}$. The MMI $I(\mathcal{X};\mathcal{Y};\mathcal{Z})$ (highlighted gray) defines the information gained by taking into account $I(\mathcal{X},\mathcal{Y};\mathcal{Z})$ instead of a pairwise combination $I(\mathcal{X};\mathcal{Z})+I(\mathcal{Y};\mathcal{Z})$. For this, the CMI $I(\cdot;\cdot|\cdot)$ determines the information shared between two processes when a third one is known.

dencies between multiple processes. One important exception not taken into account by this diagram is further explained in the following. Consider two independent processes $\{\mathcal{X},\mathcal{Y}\}$ which do not share mutual information, i.e., $I(\mathcal{X};\mathcal{Y})=0$. Regarding Figure 3.3, the CMI with another process $\mathcal{Z}$ is $I(\mathcal{X};\mathcal{Y}|\mathcal{Z}) \leq I(\mathcal{X};\mathcal{Y})$. Hence, with knowledge about $\mathcal{Z}$ the information shared between $\{\mathcal{X},\mathcal{Y}\}$ is usually reduced. In terms of information theory this is referred to as *redundancy* effect. For this, independent processes $\{\mathcal{X},\mathcal{Y}\}$ should not share information, i.e., $I(\mathcal{X};\mathcal{Y}|\mathcal{Z})=0$. Unfortunately, this is not necessarily true since the CMI, also for independent processes, can be $I(\mathcal{X};\mathcal{Y}|\mathcal{Z}) > I(\mathcal{X};\mathcal{Y})$. Broadly speaking, due to the interaction with $\mathcal{Z}$, $\{\mathcal{X},\mathcal{Y}\}$ starts to share information. Omitted by the simplified illustration of the Venn diagram, such *synergistic* effects occur when the MMI becomes negative. More precisely, the information implied by a process $\mathcal{Z}$ for two corresponding processes $\{\mathcal{X},\mathcal{Y}\}$ is interpreted with regard to the sign of the MMI

$$I(\mathcal{X};\mathcal{Y};\mathcal{Z}) = I(\mathcal{X};\mathcal{Y}) - I(\mathcal{X};\mathcal{Y}|\mathcal{Z}) = \begin{cases} \text{redundancy} & \text{for } > 0 \\ \text{no effect} & \text{for } = 0 \\ \text{synergy} & \text{for } < 0 \end{cases}. \tag{3.10}$$

As mentioned before the information shared between processes is quantified without taking into account time delayed dependencies. Hence, mutual information as well as the multivariate modification have no directional sense. This is tolerable when estimating the states of simultaneous processes, e.g., the unfair coin experiment in

which the coin is influenced by the trigger without any lag of time. Obviously, such simultaneous influences are rather the exception than the norm but give a first estimate of correlations between processes. How to detect time delayed information transfer between processes is introduced in the following section.

3.1.3 Transfer Entropy

In the previous section, it was shown that mutual information has no directional sense when determining the amount of information two processes share and therefore cannot be applied to detect delayed correlations. The general form of the corresponding Equation 3.7 is given by the *Kullback-Leibler divergence*

$$D(p||q) = \sum_{x \in \mathcal{X}} p(x) \log_a \frac{p(x)}{q(x)}. \tag{3.11}$$

where p and q are two probability distributions with the same state space $\{x_1, \ldots, x_n\} \in \mathcal{X}$ and $D(p||q) \geq 0$. Thus, the Kullback-Leibler divergence measures the amount of information lost when using q instead of p and is only zero if p equals q. Generally speaking, it is a measure of dissimilarity between two probability distributions.

To give a better understanding of the Kullback-Leiber divergence the probability distribution for a fair and an unfair coin from Section 3.1.1 are compared. On the one hand, for a fair coin the probability for head (p_{head}) or tail (p_{tail}) is both $\frac{1}{2}$ and the corresponding Shannon entropy is $H_{fair} = 1\,\text{bit}$ (see Equation 3.3 p. 27). On the other hand, a unfair coin with probabilities $q_{head} = \frac{3}{4}$ and $q_{tail} = \frac{1}{4}$ contains $H_{unfair} = 0.8112\,\text{bit}$ Shannon entropy (see Equation 3.4 p. 28). Consequently, by utilizing Equation 3.11 for the case of the unfair coin, the amount of information lost when using q to approximate values of p is

$$\begin{aligned} D(p||q) &= \left(q_{head} \cdot \log_2 \frac{q_{head}}{p_{head}} \right) + \left(q_{tail} \cdot \log_2 \frac{q_{tail}}{p_{tail}} \right) \\ &= \left(\frac{3}{4} \cdot \log_2 \frac{\frac{3}{4}}{\frac{1}{2}} \right) + \left(\frac{1}{4} \cdot \log_2 \frac{\frac{1}{4}}{\frac{1}{2}} \right) \\ &= \left(\frac{3}{4} \cdot 0.585 \right) + \left(\frac{1}{4} \cdot -1 \right) \\ &= 0.1887\,\text{bit}, \end{aligned}$$

which is equivalent to the difference of the particular Shannon entropies $D(p||q) = H_{fair} - H_{unfair} = 0.1887\,\text{bit}$.

However, an advantage of the Kullback-Leibler divergence, in contrast to the more specific mutual information, is its non-symmetric property. This is shown by swapping

p and q. The amount of information lost when utilizing p to approximate values of q is

$$\begin{aligned} D(q||p) &= \left(p_{head} \cdot \log_2 \frac{p_{head}}{q_{head}}\right) + \left(p_{tail} \cdot \log_2 \frac{p_{tail}}{q_{tail}}\right) \\ &= \left(\frac{1}{2} \cdot \log_2 \frac{\frac{1}{2}}{\frac{3}{4}}\right) + \left(\frac{1}{2} \cdot \log_2 \frac{\frac{1}{2}}{\frac{1}{4}}\right) \\ &= \left(\frac{1}{2} \cdot -0.585\right) + \left(\frac{1}{2} \cdot 1\right) \\ &= 0.2075\,\text{bit}. \end{aligned}$$

Obviously, $D(q||p)$ differs from $D(p||q)$ what proves the non-symmetric property of the Kullback-Leibler divergence.

Next, time delayed relations are taken into account by utilizing the *entropy rate* which for a process $\mathcal{X}$ with states $\{x_1, \ldots, x_n\} \in \mathcal{X}$ is defined by

$$\begin{aligned} h(\mathcal{X}) &= -\sum_{i=1}^{n} p(x_{i+1}, x_i) \log_a \frac{p(x_{i+1}, x_i)}{p(x_i)} \\ &= -\sum_{i=1}^{n} p(x_{i+1}, x_i) \log_a p(x_{i+1}|x_i), \end{aligned} \tag{3.12}$$

where $p(\cdot|\cdot)$ describes the conditional probability of two probability distributions and (x_i, x_{i+1}) are two consecutive states in a temporal sequence. In contrast to the Shannon entropy, the entropy rate determines the average amount of information which is needed to encode the next state of the system when all previous states are known. Hence, the entropy rate can be interpreted as the magnitude of how strong previous states of a process $\{x_1, \ldots, x_m\}$ are influencing its next state x_{m+1}. A process which only depends on the previous state x_m is called a *Markov process*

$$p(x_{m+1}|x_1, \ldots, x_m) = p(x_{m+1}|x_m) \tag{3.13}$$

and therefore its transition probabilities are independent from the past.

This can be generalized to multiple processes by taking into account another simultaneous process $y_1, \ldots, y_m \in \mathcal{Y}$

$$p(x_{m+1}|x_1, \ldots, x_m, y_1, \ldots, y_m) = p(x_{m+1}|x_m), \tag{3.14}$$

where $\mathcal{X}$ and $\mathcal{Y}$ share the same state space. Broadly speaking, it is assumed that process $\mathcal{Y}$ has no effect on the transition probability of process $\mathcal{X}$. This is quantified

Tab. 3.3: 20 repetitions of tossing a coin. $\mathcal{X}$ is the result of the coin (heads are zeroes and tails are ones) while process $\mathcal{Y}$ is the perfect prediction of $\mathcal{X}$ with a delay of one.

$\mathcal{X}$	0	1	1	1	0	1	0	1	0	1	0	1	1	0	0	0	1	0	1	1
$\mathcal{Y}$	1	1	1	0	1	0	1	0	1	0	1	1	0	0	0	1	0	1	1	0

Tab. 3.4: The relevant probability distributions for calculating the entropy rate and TE of the coin tossing experiment shown in Table 3.3.

$x+1$	x	y	$p(x+1,x,y)$	$p(x,y)$	$p(x+1,x)$	$p(x)$
0	0	0	$\frac{2}{19}$	$\frac{2}{19}$	$\frac{2}{19}$	$\frac{9}{19}$
0	1	0	$\frac{6}{19}$	$\frac{6}{19}$	$\frac{6}{19}$	$\frac{10}{19}$
1	0	1	$\frac{7}{19}$	$\frac{7}{19}$	$\frac{7}{19}$	$\frac{9}{19}$
1	1	1	$\frac{4}{19}$	$\frac{4}{19}$	$\frac{4}{19}$	$\frac{10}{19}$

by Transfer Entropy (TE) [93],

$$\begin{aligned} TE_{\mathcal{Y}\to\mathcal{X}} &= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x+1,x,y)\log_a \frac{\frac{p(x+1,x,y)}{p(x,y)}}{\frac{p(x+1,x)}{p(x)}} \\ &= \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x+1,x,y)\log_a \frac{p(x+1|x,y)}{p(x+1|x)}, \end{aligned} \tag{3.15}$$

which, as mutual information, is a kind of Kullback-Leibler divergence. The resulting amount of information specifies how strong the future states of $\mathcal{X}$ are influenced by the past of $\mathcal{Y}$ or more precisely the average decrease in uncertainty about the future state of $\mathcal{X}$ when $\mathcal{Y}$ is known.

For a better understanding of the relations between entropy rate and TE another coin example is given. As shown in Table 3.3 the process $\mathcal{Y}$ is a perfect predictor for the next state of process $\mathcal{X}$ which again represents the state of the coin. The corresponding probability distribution is shown in Table 3.4. Due to the perfect prediction from $\mathcal{Y}\to\mathcal{X}$ the TE is expected to be maximal. To prove this assumption, the entropy rate of the coin is calculated by utilizing Equation 3.12:

$$\begin{aligned} h(\mathcal{X}) &= -\left[\left(\frac{2}{19}\cdot\log_2\frac{\frac{2}{19}}{\frac{9}{19}}\right)+\left(\frac{6}{19}\cdot\log_2\frac{\frac{6}{19}}{\frac{10}{19}}\right)+\left(\frac{7}{19}\cdot\log_2\frac{\frac{7}{19}}{\frac{9}{19}}\right)+\left(\frac{4}{19}\cdot\log_2\frac{\frac{4}{19}}{\frac{10}{19}}\right)\right] \\ &= -\left[\left(\frac{2}{19}\cdot -2.1699\right)+\left(\frac{6}{19}\cdot -0.737\right)+\left(\frac{7}{19}\cdot -0.3626\right)+\left(\frac{4}{19}\cdot -1.3219\right)\right] \\ &= 0.873\,\text{bit}. \end{aligned}$$

Thus, an average of 0.873 bit is required to encode the next state of the coin. In more detail, the uncertainty for the prediction of the coin's next state is 0.873 bit. By

utilizing Equation 3.15, the corresponding TE from $\mathcal{Y} \to \mathcal{X}$ is calculated

$$\begin{aligned} TE_{\mathcal{Y}\to\mathcal{X}} &= \left(\frac{2}{19}\cdot\log_2\frac{\frac{2}{19}}{\frac{2}{19}\frac{2}{19}}\right) + \left(\frac{6}{19}\cdot\log_2\frac{\frac{6}{19}}{\frac{6}{19}\frac{6}{19}}\right) + \left(\frac{7}{19}\cdot\log_2\frac{\frac{7}{19}}{\frac{7}{19}\frac{7}{19}}\right) + \left(\frac{4}{19}\cdot\log_2\frac{\frac{4}{19}}{\frac{4}{19}\frac{4}{19}}\right) \\ &= \left(\frac{2}{19}\cdot 2.1699\right) + \left(\frac{6}{19}\cdot 0.737\right) + \left(\frac{7}{19}\cdot 0.3626\right) + \left(\frac{4}{19}\cdot 1.3219\right) \\ &= 0.873\,\text{bit}. \end{aligned}$$

As can be seen, the TE from $\mathcal{Y} \to \mathcal{X}$ equals the entropy rate. This is due to the fact that with knowledge about process $\mathcal{Y}$ the average magnitude of uncertainty about the prediction of the next state of $\mathcal{X}$ is eliminated. Hence, given process $\mathcal{Y}$, no additional information is required to predict $\mathcal{X}$.

In contrast to that, the next state of $\mathcal{Y}$ cannot be predicted from the actual state of $\mathcal{X}$ for all cases. Accordingly, the resulting TE differs $TE_{\mathcal{Y}\to\mathcal{X}} \neq TE_{\mathcal{X}\to\mathcal{Y}}$. This shows the non-symmetric property of TE.

However, one could argue that such information transfer is not detected if it influences another than the next state of a process, e.g., the manipulated coin experiment in Section 3.1.2 which is influenced instantaneously. In order to address arbitrary temporal divergences, a time delay d is added to the TE resulting in the equation for the so called *delayed* TE [94]

$$TE_{\mathcal{Y}\to\mathcal{X}}(d) = \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x+1,x,y+1-d)\log_a\frac{p(x+1|x,y+1-d)}{p(x+1|x)}. \tag{3.16}$$

Hence, $TE_{\mathcal{Y}\to\mathcal{X}}(1)$ is suitable to detect the information transfer for the previous example while $TE_{\mathcal{Y}\to\mathcal{X}}(0)$ can be applied to detect the dependencies between trigger and coin for the experiment given in Section 3.1.2. More precisely, by utilizing its directional sense, delayed TE is able to recognize preponed ($d < 0$), instantaneous ($d = 0$) and postponed ($d > 0$) correlations.

To further take into account the history of the involved processes, *higher order* TE [95] builds patterns from a time window of past states

$$TE_{\mathcal{Y}\to\mathcal{X}}(d,k,l) = \sum_{x\in\mathcal{X},y\in\mathcal{Y}} p(x+1,x^{(k)},y^{(l)}+1-d)\log_a\frac{p(x+1|x^{(k)},y^{(l)}+1-d)}{p(x+1|x^{(k)})}. \tag{3.17}$$

Here, k and l are the number of past states taken into account to build patterns contained in the processes $\mathcal{X}$ and $\mathcal{Y}$. However, the amount of patterns increase exponentially s^{k+l}, where s is the number of possible states. Hence, for the previous coin experiment (Table 3.3) the number of patterns is $2^{1+1} = 4$ what is also shown in Table 3.4. More precisely, higher order TE generates new states by building patterns from the original ones. In return, the increased number of states results in a clearer detection of correlations but has the drawback of an increased computational demand.

3.1.4 Summary

Based on the field of information theory, several methods for measuring dependencies between discrete random variables (processes) were introduced. As shown, correlations are identified by utilizing mutual information (see Equation 3.7 p. 29) which measures the shared amount of Shannon entropy. By further taking into account multiple conditional processes, MMI (see Equation 3.8 p. 30) allows to detect redundant and synergistic effects. Since no temporal relations are taken into account, these measurements are limited to the detection of simultaneous correlations. Time delayed correlations can be discovered by considering the corresponding entropy rates. More precisely, given its whole past, the uncertainty about the future state of a process is measured. The reduction of uncertainty resulting from the knowledge about another process is quantified by TE (see Equation 3.15 p. 34) which in contrast to mutual information has a directional sense. Utilizing different delays, TE (see Equation 3.16 p. 35) is also able to detect preponed, instantaneous and postponed correlations.

Mutual information and TE are both a kind of Kullback-Leibler divergence. Hence, as proven by Hlavkov-Schindler et al. [41], TE can be written in terms of CMI

$$TE_{\mathcal{Y}\to\mathcal{X}} = I(\mathcal{X}+1;\mathcal{Y}|\mathcal{X}). \tag{3.18}$$

Here, given the actual state of $\mathcal{X}$, the amount of information shared between the future of $\mathcal{X}$ and the actual state of $\mathcal{Y}$ is measured. This is equivalent to the definition of TE. Therefore knowing $\mathcal{Y}$ reduces the uncertainty about the future of $\mathcal{X}$ while $\mathcal{Y}$ is not necessarily a good predictor for actual states of $\mathcal{X}$.

The resulting amount of information strongly depends on the corresponding processes and therefore a uniform measure is given by

$$\begin{aligned} I_{ratio}^{(\mathcal{X};\mathcal{Y};\mathcal{Z})} &= 1 - \left(\frac{min(H(\mathcal{X}),H(\mathcal{Y}),H(\mathcal{Z})) - I(\mathcal{X};\mathcal{Y};\mathcal{Z})}{min(H(\mathcal{X}),H(\mathcal{Y}),H(\mathcal{Z}))}\right), \\ &\forall min(H(\mathcal{X}),H(\mathcal{Y}),H(\mathcal{Z})) > 0. \end{aligned} \tag{3.19}$$

Thus, $I_{ratio}^{(\mathcal{X};\mathcal{Y};\mathcal{Z})} = 0$ implies that no information is shared between the processes $\{\mathcal{X},\mathcal{Y},\mathcal{Z}\}$. In contrast to that, $I_{ratio}^{(\mathcal{X};\mathcal{Y};\mathcal{Z})} = 1$ means that they completely share the same information. Furthermore, if one of the involved processes has no Shannon entropy then no potentially shared information is contained at all.

To apply these information-theoretic measures, the corresponding processes need to share the same state space and therefore must be discrete sets. This is often not the case for real world setups which instead provide continuous time series data. Hence, the process states are discretized utilizing a so called *binning estimator*. More precisely, the underlying continuous value is estimated by quantifying the process states with a fixed number of bits what results in limited number of levels. The calculated information measures are then approximating the corresponding probability distributions [41].

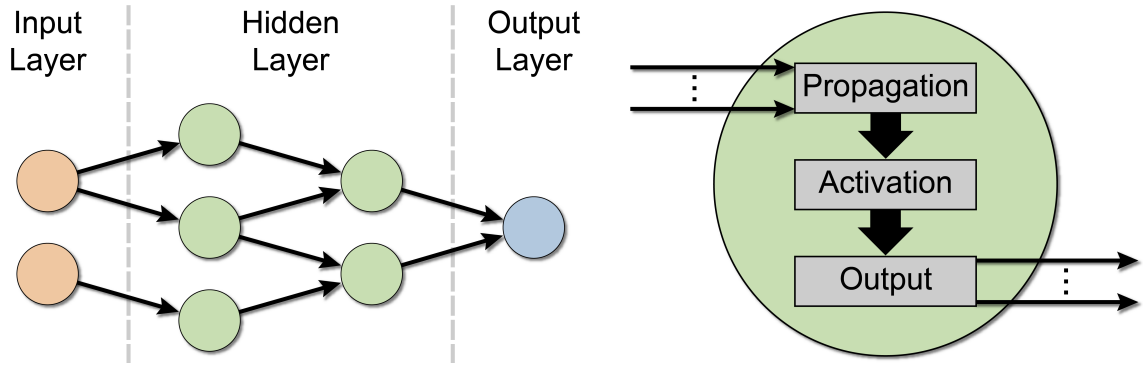


Fig. 3.4: Inspired by their biological origin, ANNs implement multiple layers of interconnected neurons. Left: An ANN is constructed of an input, up to multiple hidden and one output layer. In order to process information each of these layers contains at least one neuron. Right: A neuron's structure located in the hidden or output layer. First, the arriving information (input) is combined by the propagation function. Then, the overall stimulus is used to calculate the activation state of the neuron. The level of activation determines the amount of output information which is transferred to further connected neurons (hidden neurons) or is utilized as an estimate of the particular function (output neurons). In contrast to that, input layer variables directly receive sensory feedback and are used to feed information into the neural network.

3.2 Artificial Neural Networks

The goal of an Artificial Neural Network (ANN) is to approximate an unknown linear or non-linear function f in order to calculate the corresponding value y for a set of parameters $\mathbf{x}$

$$y = f(\mathbf{x}).$$

In a complex dynamic environment often not all parameters are given and therefore $\mathbf{x}$ is only a subset of the influencing parameters which may also contain noise. Hence, a neural network is approximating the underlying function resulting in an estimate for the corresponding value and is defined as sorted triple

$$(\mathcal{N}, \mathcal{C}, \mathbf{W}), \tag{3.20}$$

where $\mathcal{N}$ is a set of neurons and $(i, j) \in \mathcal{C}$ is a set of directed connections between two neurons from $i \rightarrow j$. Furthermore, each of these connections is weighted by the corresponding entry of a square matrix $\mathbf{W}_{i,j} \in \mathbf{W}$, where connections with zero weight does not exist. Whether excited or inhibited, the task of connections is to transfer information between the processing neurons.

More precisely, as illustrated in Figure 3.4 left, such networks are commonly divided into three kind of layers. The first is an *input layer* $\mathcal{I} \in \mathcal{N}$ which is containing values of the available variables. In general, these inputs can be understood as the sensory feedback that is representing the state of the environment. Next, the information is transferred to the processing layers $\mathcal{H} \in \mathcal{N}$. These layers are representing the approximated function and are usually referred to as *hidden layers*. This is due to the fact that they are not directly accessed from the outside. Instead they are influenced by the particular learning technique. Finally, the result of the approximated function

is returned by output neurons contained in the *output layer* $\mathcal{O} \in \mathcal{N}$.

Such neural networks are inspired by the principles of biological neural networks, e.g., the CNS of humans, and are well suited to give an estimate about a situation also when some parameters are unknown or affected by noise. This is due to their ability to remember familiar situations and generalize from experience. In order to implement such learning capabilities also ANNs require a training phase. For this, similar to their biological origin, training data is utilized to adjust the network weights accordingly. However, the training mechanism strongly depends on the particular topology and will be clarified later. First of all, the basic principles of artificial neurons and different types of connections are introduced.

3.2.1 Neurons

The human brain consists of about 86 billion neurons [96]. These biological neurons have a complex structure and high interaction capabilities which are not practicable for the implementation on state-of-the-art computational systems. Hence, they are drastically simplified to the structure illustrated in Figure 3.4 right. This reduced model is processing multiple input values from other neurons utilizing the following successive pattern:

1. propagation function
2. activation function
3. output function

These functions are applied to each neuron except for variables contained in the input layer. In particular, (1) is used to transform the information received from other neurons to an adequate input. Stimulated by these inputs, (2) represents the internal activation state of a neuron. Finally, (3) generates a neuron's output which again can be used as input for other neurons. Below, each component is explained in detail.

Propagation Function

Biological neurons can have multiple input connections from other neurons which are accumulated to one stimulus. Similar to this, artificial neurons have a vectorial input $\mathbf{x} = (x_1, \ldots, x_n)$. The network input for the j-neuron is then determined by the accumulated weighted sum

$$inp_j = \sum_{i=1}^{n} x_i \mathbf{W}_{i,j}, \tag{3.21}$$

where $\mathbf{W}_{i,j}$ is the weight of the connection from $i \to j$. Depending on the particular task also other functions, e.g., minimum, maximum or product can be useful. However, the resulting network input is influencing the internal state of the neuron which is determined by the following activation function.

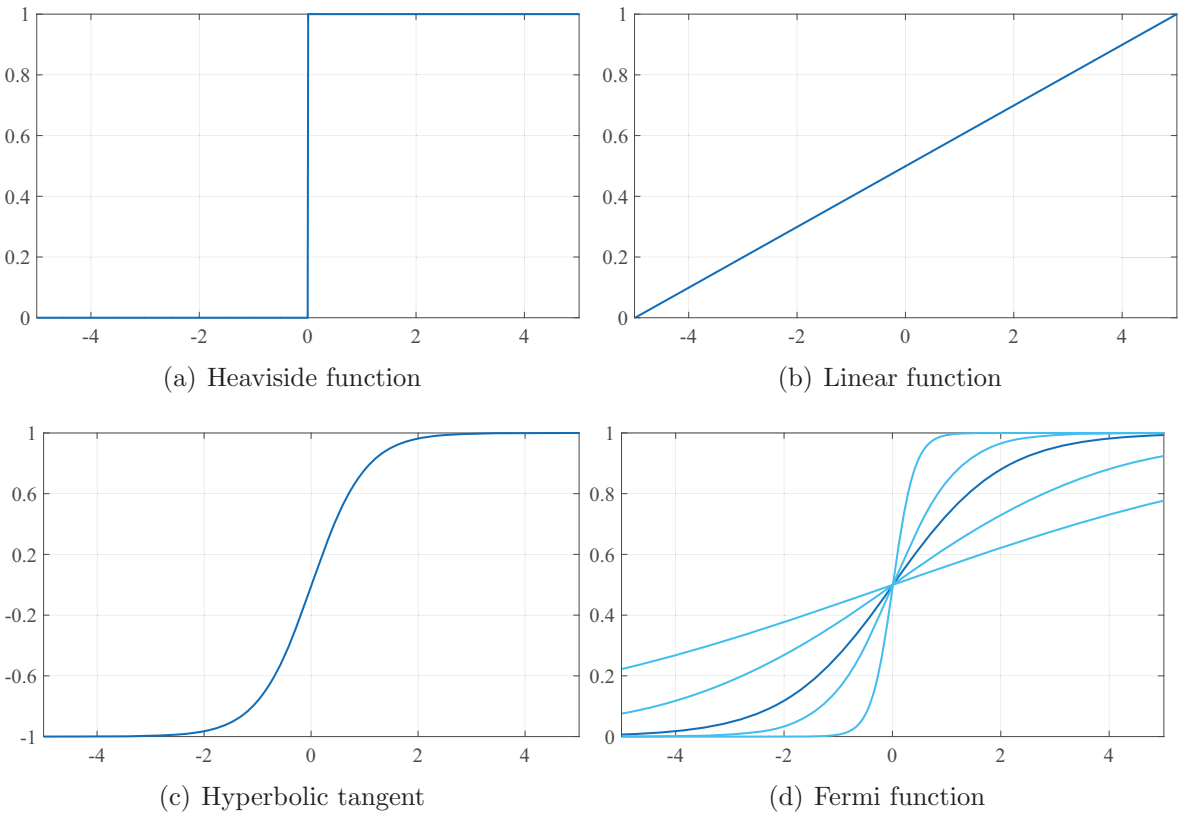


Fig. 3.5: The activation function f_{act} transforms a neuron's combined input inp to an activation state act. Depending on the particular function, this state has its inflection point at a specific threshold Θ. Here, the binary Heaviside function is non-differentiable at Θ and therefore is not suitable for a gradient based learning technique. To approximate its functionality a parametrized sigmoidal function called Fermi function is applied.

Activation Function

In biology, a neuron has an activation state which depends on the overall external stimulus and starts firing after the potential reaches a threshold. In contrast to that, a usual artificial neuron generates always an output which strongly depends on its activation value. For this, the activation function f_{act} maps a neuron's accumulated inputs inp_j to the activation value

$$act_j = f_{act}(inp_j, \Theta_j), \tag{3.22}$$

where Θ_j is an adjustable threshold defined by the maximum gradient of the corresponding activation function. Typically, the activation value of a neuron reacts strongest close to the particular threshold of the activation function. Figure 3.5 illustrates various popular activation functions. The most simple one is a binary threshold function which is also called *Heaviside-function* (Figure 3.5(a)) that has two possible activation states $0, 1$ switching at Θ_j (usually $\Theta_j = 0$). As a matter of fact, this function is not differentiable at Θ_j which is a major disadvantage for the later introduced learning technique. In contrast to that, a linear function as illustrated in Figure 3.5(b) is differentiable. The main drawback of linear functions is that, for multiple layers,

they can be composed to only one function. Therefore, they are inappropriate for neural networks with more than one layer of adaptable weights. To solve this, more common functions are utilizing the hyperbolic tangent $tanh(inp_j)$ which, as shown in Figure 3.5(c) is in a range of $(-1, 1)$. Furthermore, parameter dependent sigmoidal functions as the so called *Fermi-function* illustrated in Figure 3.5(d)

$$\frac{1}{1 + \mathrm{e}^{\frac{-inp_j}{T}}},$$

where T is an additional 'temperature' parameter, are used to achieve an adaptable behavior in a range of $(0, 1)$. Here, $T = 1$ equals the usual sigmoid function (highlighted dark blue) while $T < 1$ gets closer to the usage of a Heaviside-function and $T > 1$ is more similar to a linear function. Depending on the chosen activation function, the resulting activation value is utilized to calculate an appropriate output as follows.

Output Function

The output function f_{out} makes use of the neuron's activation value act_j to determine the output value

$$out_j = f_{out}(act_j), \tag{3.23}$$

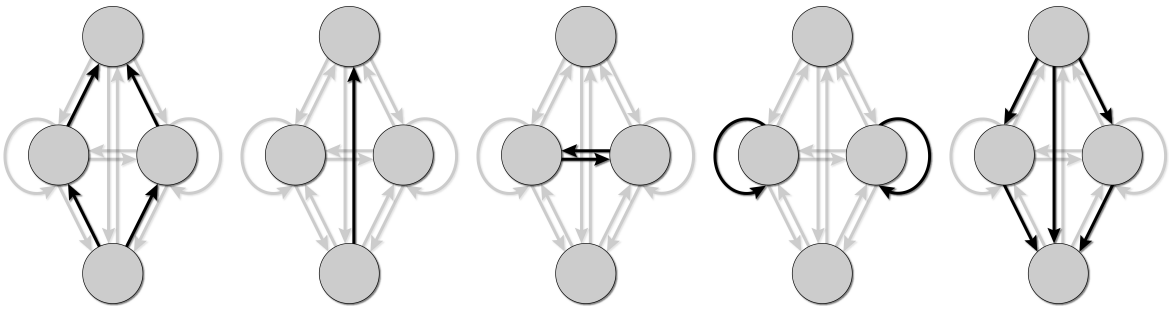
which represents the final output of a neuron and is transferred to all connected output neurons. Such an additional mapping of the activation value is relevant if the range of the activation function is not sufficient for a desired output. However, in practice the output function is often the identity of the activation function $f_{out}(act_j) = act_j$.

How single neurons are combined to a neural network depends on their particular connections where the resulting network topology influences the efficiency of learning as well as the quality of the results. Hence, different types of connections are examined in the following section.

3.2.2 Connections

As previously stated, the human brain contains about 86 billion neurons which need to transfer information at a certain level of activation. For this, each neuron has an average of 1000 synapses resulting in a minimum of 43 trillion connections. The underlying complex biological structure of these synapses is simplified by the already defined connections $\mathcal{C}$. Figure 3.6 illustrates the most relevant types of connections. Most familiar are *direct connections* which are restricted from one layer to the next one. If one layer is skipped this is referred to as second order direct connections or *shortcut connections*. Connections within a layer are called *lateral connections* and can cause loops which consequently lead to recurrence. Broadly speaking, recurrence means that a neuron is influenced by its own past. Hence, the highest level of recurrence is direct recurrence which is generated by *self-connections*. Also *indirect connections* towards a previous layer can result in recurrence.

Depending on the permitted connection types different network topologies can be



(a) Direct (b) Shortcut (c) Lateral (d) Self (e) Indirect

Fig. 3.6: ANNs transfer information by different types of connections which specify their topology. For example, only direct and shortcut connections are allowed for the construction of FNNs. In contrast, RNNs have no such restrictions and therefore are able to remember past information where recurrences are implied by lateral, self and indirect connections.

constructed. The most popular are the so called *feed forward* and *recurrent* network architectures. On the one hand, the FNN utilizes only direct and shortcut connections and but does not allow recurrences. Hence, an FNN cannot remember its past and consequently predict values independent from its previous states. On the other hand, an RNN utilize recurrent connections which enable it to remember previous inputs in its internal states. Therefore, the output of an RNN is influenced by its past. As will be shown later, RNNs usually outperform the estimation accuracy of FNNs but are more complex to learn. The information flow between two neurons further depends on the weight of a particular connection. Here, a large weight increases the transmitted information while small ones inhibit the information flow. The learning ability of a neural network is then realized by adjusting the overall connection weights $\mathbf{W}$.

3.2.3 Learning

In order to learn a prediction of the actual output of a system, an ANN is trained by a set of training samples $s \in \mathcal{S}$. Such ground truth data contains pairs of training samples $s = (\mathbf{in}, \mathbf{out})$, where $\mathbf{in} = (in_1, \ldots, in_n)^\mathsf{T}$ is the input data and $\mathbf{out} = (out_1, \ldots, out_m)^\mathsf{T}$ is the corresponding output. This is similar to the definition of supervised learning (see Section 2.1) and enables the resulting neural network to generate continuous estimates for arbitrary inputs. Hence, $\mathbf{in}$ is utilized to initialize the input layer $\mathcal{I}$ while $\mathbf{out}$ is the desired correct output. Furthermore, the connection weights $\mathbf{W}$ between these layers are randomly initialized. Then, a sample's input data $\mathbf{in}$ is propagated forward through the neural network resulting in an estimated output $\hat{\mathbf{out}}$ contained in the output layer $\mathcal{O}$. The difference between the correct output and those estimates yields the estimation error

$$E_s = \mathbf{out} - \hat{\mathbf{out}} = \begin{pmatrix} out_1 - \hat{out}_1 \\ \vdots \\ out_m - \hat{out}_m \end{pmatrix} \tag{3.24}$$

and is utilized as quality criterion of the corresponding neural network. In order to minimize E_s, the neural network's connection weights $\mathbf{W}$ and the neurons threshold values $\mathbf{\Theta}$ need to be adapted. To unify the adaptation process, the latter is realized by utilizing *bias neurons* [97, p. 43–45]. More precisely, neurons receive an additional input from a bias neuron whose output value is always one while the connection between both neurons is weighted utilizing the neuron's negative threshold value $-\Theta$. As a result, threshold values can be handled like connection weights and therefore are usually omitted.

The adaptation of the connection weights can be done *online* for each training sample separately or *offline* for all training samples at once. Also splitting the training samples into *batches* is a common technique. The network is learning by subsequently adapting the connection weights and therefore increasing the estimation accuracy. Here, each adaptation of the network's connection weights is called an *epoch*. In the following, the ability of neural networks to learn from experience is explained in more detail.

Each training sample s yields an estimation error as determined in Equation 3.24. The learning curve specifies the evolution of this error over a period of epochs and therefore is an indicator for the network's progress. As mentioned above, the training samples can be processed online, offline or in batches. Hence, also the estimation error for one epoch is calculated depending on the particular partitioning. In case of online learning the weights are adapted based on a single training sample $s = (\mathbf{in}, \mathbf{out})$ while, for offline learning, the weights are adapted after all training samples $\mathcal{S}$ are processed by the neural network. The total error is then accumulated by the particular error function $E_{\mathcal{S}} = \sum_{s \in \mathcal{S}} E_s$. Similar to offline learning, a subset of training samples $\mathcal{S}_b \in \mathcal{S}$ is used to calculate the error for batch learning $E_{\mathcal{S}_b} = \sum_{s \in \mathcal{S}_b} E_s$.

There are three error functions which are commonly used to calculate the estimation error. Usually, the Mean Squared Error (MSE) is utilized since it requires less computational effort than the Euclidean Distance (EUD). Furthermore, the Root Mean Square Error (RMSE) is a popular solution to take into account outliers more accurately.

$$E_s^{MSE} = \frac{1}{m} \sum_{i=1}^{m} (out_i - \hat{out}_i)^2 \tag{3.25}$$

$$E_s^{EUD} = \sqrt{\sum_{i=1}^{m} (out_i - \hat{out}_i)^2} \tag{3.26}$$

$$E_s^{RMSE} = \sqrt{\frac{\sum_{i=1}^{m} (out_i - \hat{out}_i)^2}{m}} \tag{3.27}$$

The resulting error values can be understood as a normalized measure of distance between estimated and correct output.

Another error function builds on the Kullback-Leibler divergence $D(p||q)$ which was introduced in the previous section (see Equation 3.11 p. 32). Here, q is the probability distribution of the estimated output values while p is the correct probability distribution determined from the training samples. The additional amount of bits required to encode the training samples with q instead of p is then used to evaluate the network's

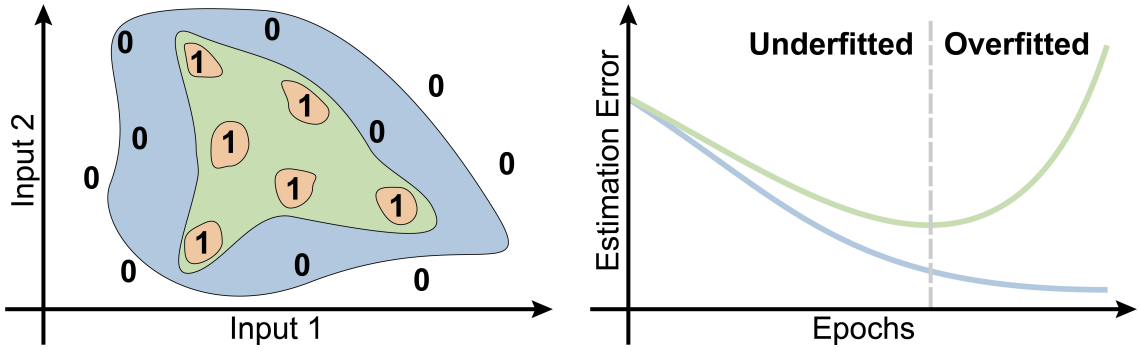


Fig. 3.7: Estimating accurate outputs for arbitrary inputs is a desirable strength of neural networks. Left: An ANN is learned for a simple classification task with two input variables. After learning a few epochs the network is still underfitted which results in erroneous classifications (blue area). The classification after a sufficient number of epochs contains all classes correctly while the green area shows its ability to generalize also for novel inputs between them. After continuing learning for numerous epochs, the ANN is overfitted and perfectly memorizes the given training classes (orange area) but lost its ability to generalize. Right: In order to stop learning before memorizing begins, the data set is divided into training and validation samples. Training samples are utilized by the learning algorithm (blue curve) while validation samples are not influencing the weight adaptation. Instead they are used to evaluate the network's ability to generalize outputs for unknown data (green curve). An increasing estimation error for validation samples indicates that learning swapped from under to overfitting and therefore the network starts memorizing the training samples.

estimation accuracy. In particular, the Cross Entropy Error (CEE) is determined by

$$E_s^{CEE} = H(\mathcal{S}) + D(p||q), \tag{3.28}$$

where $H(\mathcal{S})$ denotes the Shannon entropy (see Equation 3.2 p. 27) of all training samples. One drawback of this error function is that (as stated in Section 3.1.1) the Shannon entropy can only be calculated for discrete sets of data. Hence, CEE functions are commonly applied to various kind of classification problems.

During training, the error function is applied to stop learning at a certain level of accuracy or generalizability. Estimating accurate outputs for arbitrary inputs is a desirable strength of neural networks. As mentioned earlier this is close to the idea of approximating a systems output or function for previously unknown inputs. Hence, the goal of the learning process is not to exactly reproduce the training data but instead to approximate the underlying function. This ability is corrupted when the training is done with homogeneous data for too many epochs since the weights may be perfectly adapted to the particular training samples. In more detail, also small changes of so far unknown input data may result in strong fluctuations and wrong estimation results. The corresponding effect of perfectly memorizing the training samples, while losing the ability to generalize outputs for arbitrary inputs, is also known as *overfitting*. Figure 3.7 left shows the output of an overfitted compared to a well-trained neural network for a simple classification task. As can be seen, the overfitted network memorizes the given examples very accurately but loses its ability to generalize for inputs which are unknown during training.

In order to stop learning before memorizing begins, the training samples $\mathcal{S}$ are commonly divided into two disjoint sets $\{\mathcal{S}_t, \mathcal{S}_v\}$. As before, the samples in $\mathcal{S}_t$ are used to subsequently optimize the connection weights while samples in $\mathcal{S}_v$ are used to validate the estimation accuracy. Hence, data which is not involved in the learning process is utilized to evaluate the network's estimation accuracy. For this, $\mathcal{S}_v$ is propagated forward through the network after each epoch. This results in an estimation error which is comparable to the error produced by $\mathcal{S}_t$.

The learning curves for both, training (blue curve) and validation samples (green curve), are illustrated in Figure 3.7 right. As can be seen, the estimation accuracy of the validation samples is usually worse than for the training samples. This is due to the fact that deviations contained in the validation data are not considered by the learning process. The network is called *underfitted* as long as the error for both sets of samples decreases and therefore the estimation accuracy can still be increased. When the error of the validation samples increases it can be assumed that the neural network starts losing its ability to generalize for arbitrary inputs and instead begins to memorize the training samples. Hence, the learning procedure should be stopped when overfitting begins or even earlier, e.g., after reaching a predefined number of epochs or achieving a sufficient level of accuracy.

However, the core component to generate an accurate and generalizing neural network is the adequate adaptation of its connection weights. To solve this, the *backpropagation of error* technique is introduced in the following section.

Backpropagation of Error

The basic idea to optimize a neural network is realized by adapting its connection weights. For this, the connection weights are randomly initialized before the first epoch. This has the advantage that the neural network always has different but similar initial solutions. As will be shown later, this is a major advantage when searching for an optimal solution.

To keep it simple, a classical Single-Layer Perceptron (SLP) with input, output but no hidden layers and a linear activation function is utilized in the following. Such a neural network contains exactly one layer of connection weights that are stored in a the vector $\mathbf{w}$. In contrast to that, the input and output neurons as well as the connections between them are predefined and do not change during training. Consequently, the particular error function $E(\mathbf{w})$ is influenced by the network's weight configuration only. In order to minimize $E(\mathbf{w})$, the gradient descent procedure is applied. Hence, the weights are adapted with regard to the gradient of the error function $\nabla E(\mathbf{w})$. This procedure is independent from the dimensionality of the function and therefore can be applied to ANNs with an arbitrary number of connections. More precisely, for each epoch, the magnitude of the network's weight configuration is calculated by

$$\Delta \mathbf{w} = -\eta \nabla E(\mathbf{w}), \tag{3.29}$$

where η is a constant of proportionality which is also referred to as *learning rate*.

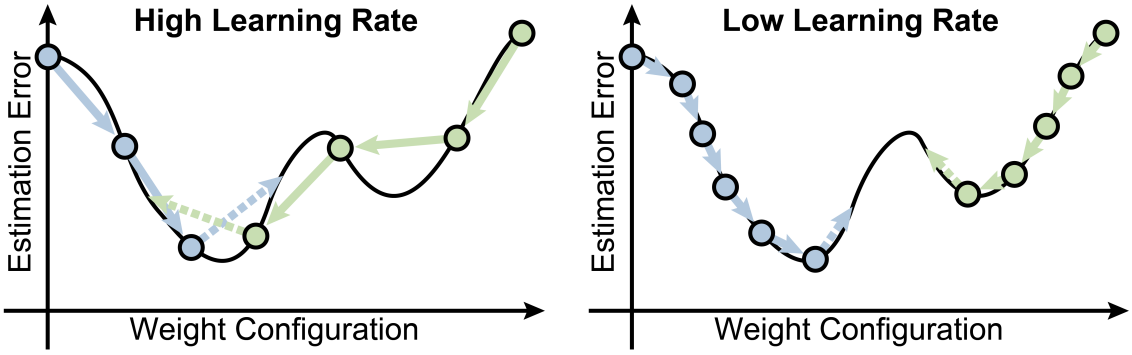


Fig. 3.8: The gradient descent technique is demonstrated for different learning rates and varying initial weight configurations (green/blue path). Left: High learning rates have a low probability of getting stuck in local minima but are less accurate. Right: Low learning rates are more accurate but have a higher probability of getting stuck in local minima. Consequently, the gradient descent technique does not guarantee to find the global minimum and depends on the particular learning rate and the initial weight configuration.

In particular, η can be used to balance between speed and accuracy of the learning procedure.

Figure 3.8 illustrates the gradient descent approach for different learning rates and one adaptable weight starting from varying initial configurations. On the one hand, high learning rates increase the search speed at the cost of decreasing accuracy. As illustrated in Figure 3.8 left, independent from the initial weight, for a large η the learning process is not able to get closer to the global minimum. On the other hand, as illustrated in Figure 3.8 right, a small η finds the global minimum but, depending on the initial weight, risks to remain in the local one. Here, the learning procedure has to be repeated more frequently which increases the computational effort. Hence, both methods are not guaranteed to find the global minimum. To solve this, dynamic learning rates combine the advantages of both by iteratively decreasing the learning rate during training. More precisely, η starts with a large value and is decreased several times during the learning procedure.

In order to adapt each weight separately, instead of the gradient of all weights $\Delta\mathbf{w}$, the first partial derivative for a single weight ∂w is utilized

$$\Delta w = -\eta \frac{\partial E(\mathbf{w})}{\partial w}. \tag{3.30}$$

Broadly speaking, every weight is adapted with regard to its influence on the error function. For offline learning, which considers all given training samples $s \in \mathcal{S}$ during one epoch, the corresponding magnitude of change is calculated by

$$\Delta w_{\mathcal{S}} = -\eta \sum_{s \in \mathcal{S}} \frac{\partial E_s(\mathbf{w})}{\partial w}. \tag{3.31}$$

Utilizing linear activation functions further allows to apply the computationally less

extensive *delta rule* which is also known as *Widrow-Hoff rule* [98]

$$\begin{aligned}\Delta w_{\mathcal{S}} &= \eta \sum_{s \in \mathcal{S}} inp_s \cdot (out_s - \hat{out}_s) \\ &= \eta \sum_{s \in \mathcal{S}} inp_s \cdot \delta_s.\end{aligned} \tag{3.32}$$

Here, the weights are adapted regarding the amount of input data inp_s and compared to the difference between correct and estimated output $\delta_s = (out_s - \hat{out}_s)$. This procedure can also be applied to online or batch learning by utilizing only a single sample or a subset of $\mathcal{S}$.

So far only neural networks with one layer of adaptable connection weights $\mathbf{w}$ are taken into account. As proven by Winder [99] the functionality of such SLPs is restricted to a subset of linearly separable functions. In contrast to that, a Multi-Layer Perceptron (MLP), which contains additional hidden layers, is able to solve more complex tasks. In particular, a MLP with two hidden layers and consequently three layers of adaptable weights is able to classify sets of any form and therefore is also termed *universal function approximator* [100]. In order to extend the concept introduced in Equation 3.32 to an arbitrary number of hidden layers, the network's weight configuration is represented by a square matrix $\mathbf{W}$. As mentioned before, each connection between neurons $i \rightarrow j$ is weighted by the corresponding entry $\mathbf{W}_{i,j} \in \mathbf{W}$. In contrast to SLPs, a MLP contains *inner neurons* $h \in \mathcal{H}$ which are connected to neurons of the previous layer $k \in \mathcal{K}$ and the following layer $l \in \mathcal{L}$. Hence, adapting the weight of an inner neuron connection is influencing subsequent neurons which are connected directly or indirectly. Additionally, MLPs contain output neurons which are only connected to neurons of the previous layer $k \in \mathcal{K}$ and, similar to SLPs, do not influence any further neurons. For any monotonous and differentiable activation function, both cases are taken into account by utilizing the delta rule resulting in

$$\Delta w_{k,h} = \eta \sum_{s \in \mathcal{S}} \hat{out}_{s,k} \cdot \delta_{s,h}, \tag{3.33}$$

where $\hat{out}_{s,k}$ is the output of the previous neuron k and therefore a partial amount of the input of the actual neuron h. Depending on whether the learning process is dealing with inner or output neurons $\delta_{s,h}$ distinguishes between

$$\delta_{s,h} = \begin{cases} \hat{out}'_{s,h} \cdot (out_{s,h} - \hat{out}_{s,h}) \text{ (output)} \\ \hat{out}'_{s,h} \cdot \sum_{l \in \mathcal{L}} (\delta_{s,l} \cdot w_{h,l}) \text{ (inner)} \end{cases}, \tag{3.34}$$

where $\hat{out}'_{s,h}$ is the first derivative of the neuron's actual output. More precisely, the magnitude of change $\Delta w_{k,h}$ for the weight of a connection between the neurons $k \rightarrow h$ is adapted with regard to

1. the learning rate η
2. the output of the previous neuron $\hat{out}_{s,k}$

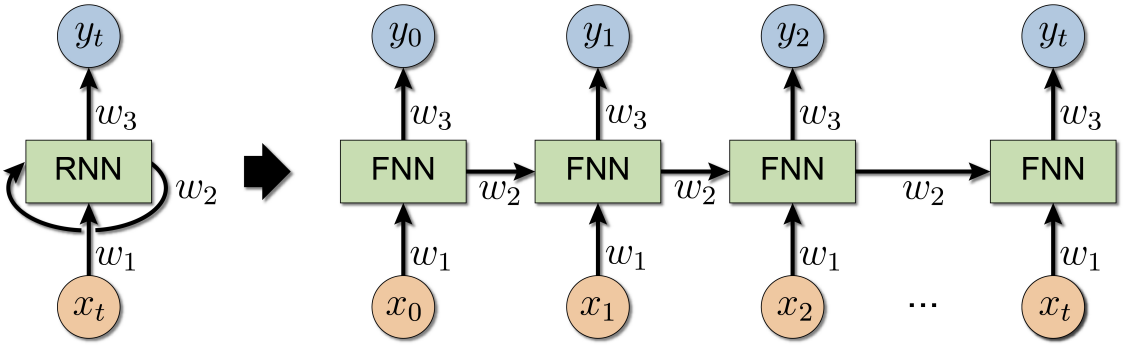


Fig. 3.9: Unfolding in time is utilized to allow the backpropagation through time algorithm being applied to RNNs. For this, each recurrent connection is replaced by the network before its connection. The multiple instances of the network are then traded as FNN where each layer share the same weight configuration. The resulting architecture is truncated after a predefined number of recurrent repetitions t to prevent the generation of an infinite deep FNN.

3. a) (output neurons) the change of activity within the particular neuron compared to its estimation error $\delta_{s,h} = \hat{out}'_{s,h} \cdot (out_{s,h} - \hat{out}_{s,h})$

   b) (inner neurons) the change of activity within the particular neuron compared to its influence on the estimation error by accounting for the change of activity of any subsequently connected neurons $\delta_{s,h} = \hat{out}'_{s,h} \cdot \sum_{l \in \mathcal{L}} (\delta_{s,l} \cdot w_{h,l})$.

This generalization of the delta rule is also referred to as *backpropagation of error* [101] and can only be utilized for monotonous and differentiable activation functions. Hence, binary threshold functions such as the Heaviside-function and other non-differentiable activation functions are not supported. In order to approximate the functionality of such functions, the Fermi-function with an appropriate temperature parameter T can be used.

As shown for the different learning rates (Figure 3.8), gradient descent and therefore the backpropagation technique is not guaranteed to find the global minimum. This problem is addressed by repeating the learning procedure multiple times with randomly chosen initial weight configurations $\mathbf{W}$. Hence, the gradient descent algorithm starts from different position which results in a variety of solutions. This process can be adapted with regard to the available time, computational power and the required accuracy but is still not guaranteed to find the optimal solution.

One problem of the backpropagation procedure is that recurrent connections would result in an infinite loop. To avoid this, the network is usually *unfolded in time* [63] and truncated after a predefined number of repetitions t. As a result, the short-term memory of the corresponding RNN is limited to a sequence of t time steps. In more detail, recurrent connections are replaced by a network which is similar to the network before the connection and therefore is one instance of the original network. As illustrated in Figure 3.9, the resulting network consists of multiple copies of the original one where each one passing its output to the particular successor. This architecture is similar to a large FNN with the difference that each layer share the same weight configuration. Hence, a slightly adapted procedure, the so called *backpropagation through*

time, is applied for learning unfolded RNNs. For a detailed description, the interested reader is referred to [102].

However, a major drawback of both backpropagation procedures is that layers closer to the input layer are learning slower than later ones. This is due to the fact that the gradients of the activation functions are multiplied with the ones of further connected neurons. For instance, the gradient of the hyperbolic tangent is $tanh(inp_j)' \leq 1$ and for a non-parametrized sigmoid function $\frac{1}{1+e^{-inp_j}} \leq 0.25$. This results in an exponential decrease of the error signal while gradients bigger than one are flipping this phenomenon. The corresponding effects are referred to as *vanishing/exploding gradient problems* [103] which increase the probability of getting stuck in a local minimum. Due to the increased number of layers, these problems are even intensified for unfolded recurrent networks. For example, vanishing gradients reduce the influence of past inputs on the error signal which prevents the network from learning long-term dependencies.

3.2.4 Long Short-Term Memory

As stated above, the vanishing/exploding gradient problem is a major disadvantage of the proposed gradient based learning technique. Hence, time delayed dependencies between input and output cannot be learned efficiently by the introduced network architectures. This problem is taken into account by the concept of the Constant Error Carrousel (CEC) which, as proven by Hochreiter [103], enforces constant error flow at any time t

$$f'_{act}(inp_j(t), \Theta_j(t))\mathbf{W}_{j,j} = 1.0, \tag{3.35}$$

where $\mathbf{W}_{j,j}$ is the weight of a recurrent self-connection to the activation function f_{act} which is also an identity function. Consequently, a linear identity function with a derivative of one is required. Furthermore, the weight of the self-connection needs to be fixed to one and is not adaptable at all. As a result, the gradient does remain constant and does not vanish or explode.

Based on this constant error flow, Hochreiter and Schmidhuber developed the Long Short-Term Memory (LSTM) [83]. Such LSTM neural networks are explicitly designed to retain information for long time periods and release it when relevant. Similar to classical neural networks, the LSTM maps data from an input to an output layer, with the difference that the hidden layers are constructed by memory blocks. As illustrated in Figure 3.10, these blocks are significantly extended versions of the previously introduced neurons. Here, the i-th memory block consists of j cells where each block contains the following functional elements:

- $f_g(inp^g_{ij})$: The cell inputs inp^g_{ij}, which are derived from the neural network's input layer or another memory cell's output, are utilized by a sigmoidal activation function f_g.

- c_{ij}: The internal state of a cell is determined by a linear activation function and has a self-recurrent connection with weight $\mathbf{W}_{j,j} = 1.0$ (the central concept of the CEC).

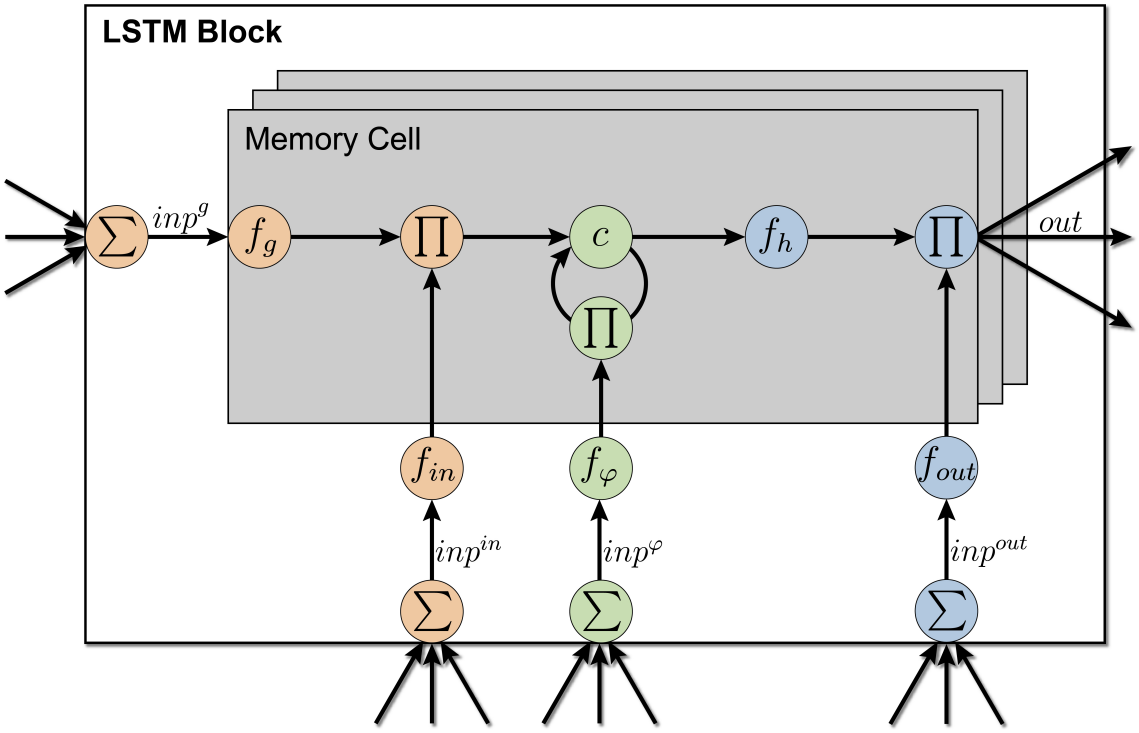


Fig. 3.10: An LSTM memory block consists of one or multiple memory cells and an input, forget and output gate which control input (orange), memorizing (green) and output (blue) information flow. By utilizing the constant error flow technique, such cells are able to remember information indefinitely unless the forget gate f_φ resets it. For this, the internal state of a cell c is implemented by a linear identity function with a constant recurrent self-connection. Furthermore, the input gate prevents the internal state from getting perturbed by useless inputs f_g while the output gate f_{out} can prevent other cells, blocks or neurons from getting influenced by this cell's output f_h. The cell's output *out* can be connected to arbitrary cells, blocks or neurons.

- $f_h(c_{ij})$: The output of a cell primarily depends on its internal state c_{ij} which is utilized by a sigmoidal activation/identity function f_h before possibly getting transmitted to another cell or output neuron.

The information processing for each cell contained in one block is further influenced by three *gates* which utilize the inputs $inp_i^{in}, inp_i^{\varphi}, inp_i^{out}$ and sigmoidal functions:

- $f_{in}(inp_i^{in})$: The input gate protects the cell states from being perturbed by irrelevant inputs.
- $f_\varphi(inp_i^{\varphi})$: The forget gate [104] is able to reset the internal cell states.
- $f_{out}(inp_i^{out})$: The output gate prevents the cells from transmitting irrelevant outputs.

Each block consists of a number of cells which share input, output and forget gate. Hence, cells contained in the same block are written, erased or read at once. As stated above, each cell includes a CEC which allows to indefinitely maintain information in its cell state c_{ij} except the forget gate resets it. This allows utilizing recurrent connections without the need of an unfolding procedure and solves the vanishing/exploding gradient

problem. Furthermore, the stored information is protected from irrelevant inputs by the input gate and forwarded through the output gate after an arbitrary lag of time.

The internal cell state is updated for discrete time steps t

$$c_{ij}(t) = f_{\varphi}(inp_i^{\varphi}(t))c_{ij}(t-1) + f_{in}(inp_i^{in}(t))f_g(inp_{ij}^{g}(t)), \tag{3.36}$$

where $c_{ij}(0) = 0$. The cell output is then determined by its internal state and the inputs to the output gate

$$out_{ij}(t) = f_h(c_{ij}(t))f_{out}(inp_i^{out}(t)). \tag{3.37}$$

Hence, when utilizing the backpropagation algorithm (see Equation 3.34 p. 46), the fixed gradients cannot vanish or explode while remembering information for unlimited periods of time. Since its introduction, the LSTM architecture was frequently extended and actually exists in several slightly different variants. However, as proven by Greff et al. [105] none of them improve the introduced standard approach and therefore are not further considered by this thesis.

3.2.5 Summary

This section introduced the biological inspired concept of ANNs. Here, the simplified idea of neurons which gather information (propagation function) and in turn transfer their state (activation function) to other neurons is the core concept to approximate linear and nonlinear functions.

For this, a set of training data is used to adapt the connection weights by a gradient based learning technique called backpropagation. Consequently, the estimation error of the entire network is iteratively minimized and therefore gets closer to the underlying function of the training data. This procedure is stopped after a predefined number of epochs, when reaching a certain level of accuracy or when overfitting occurs. The latter can be avoided by monitoring the network's generalizability with an additional set of validation data. The neural network is further enabled to remember past information by utilizing recurrent connections. For training, such RNNs are unfolded in time which increases the number of layers and consequently the computational demand.

Furthermore, depending on the particular number of layers the vanishing/exploding gradient problem occurs. As a result, the overall optimization risks to get stuck in a local minimum and therefore RNNs are usually limited to the usage of short-term dependencies. To avoid this, the concept of constant error flow and its state-of-the-art implementation called LSTM was presented. In contrast to classical RNNs, these LSTMs utilize blocks which are capable of extending, retrieving, erasing or remember information for indefinite periods of time.

3.3 Conclusion

The approach presented in the next chapter builds on the introduced mathematical fundamentals of information theory and neural networks. Here, a robot executes a task under well-known contextual conditions while recording its sensor readings. The resulting training data contain the robot's behavior-specific proprioception where a subset of the proprioception, behavior parameters or context conditions can be used as learning target when generating BSPMs.

For this, behavior-specific sensor selection is applied based on correlations between the proprioceptors and the learning target by three different information measures. (1) CMI measures the information shared between a proprioceptor and the learning target under the condition that other proprioceptors are already known. (2) MMI allows to detect synergistic and redundant effects on the information shared between proprioceptors and the learning target. (3) TE can be used as a measure of predictability from the proprioceptors to the learning target and vice versa. This allows detecting proprioceptors which are important for the behavior execution in a specific context. These proprioceptors can then be utilized to generate BSPMs by different machine learning techniques, i.e., lazy and eager learning.

This chapter introduced the mathematical background of ANNs which are constructed of neurons and weighted connections. These networks aim to approximate the function which maps proprioceptive inputs to target outputs. For this, learning is achieved by backpropagation of error which iteratively adapts the connections weights with regard to the estimated and correct output of the training data. Usually, this is repeated until the approximation is close to the underlying function of the target. As will be shown, the efficiency and accuracy of this learning procedure benefits from selecting strongly correlated proprioceptors and achieves a precision comparable to experienced workers. In the context of this thesis, the ability to continuously perform sensor regression and classification tasks in constant time is a further advantage of ANNs over other model learning techniques.

As a result, the presented BSPM approach learns accurate force estimates from prior experience only. This enables different robot platforms to adapt their behavior during human-robot collaboration, tool-usage and state classification tasks.

4 A Machine Learning Approach: Behavior-Specific Proprioception Models

The sense of self or proprioception enables humans to distinguish self-generated forces from perturbations which originate from the surrounding environment. This chapter introduces a novel machine learning concept which enhance robots with Behavior-Specific Proprioception Models (BSPMs). These models are used for force estimation and therefore find applications in human-robot collaboration, tool-usage and exploration of unstructured environments.

4.1 Behavior-Specific Proprioception Models

A BSPM establishes a link between behavior execution and the expected proprioception. By utilizing knowledge about the behavior-specific sensory-motor integration of the robot, BSPMs can be used to

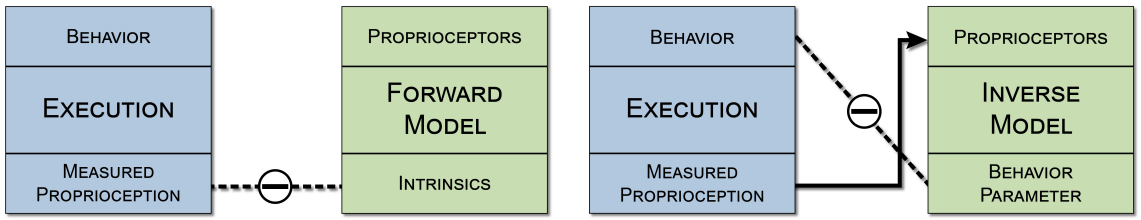
1. given high-level behavior parameters, predict the expected proprioception for unperturbed behavior execution (forward mode)

2. estimate high-level behavior parameters from measured proprioceptions, possibly obtained under perturbed execution conditions (inverse mode)

3. augment a robot's proprioception (virtual sensor mode).

By comparing the actual proprioception with the behavior-intrinsic expected proprioception, BSPMs can further be used to estimate the presence and amount of extrinsic perturbations. In the following, *intrinsics* denotes the expected sensor evolution which is induced by a regular behavior execution. In contrast, *extrinsics* is the summarizing term for influences which originate from unexpected changes in the environment. Hence, the extrinsics can be estimated by the difference between the intrinsics and the actual sensor evolution. Inspired by principles of human proprioception (see Section 1.1.2 p. 4), different BSPM modes are proposed.

(1) A Behavior-Specific Proprioception Model in Forward Mode (F-BSPM) is utilized to predict the expected proprioception. While performing the behavior, the predicted intrinsics is subtracted from the measured proprioception. This allows extracting the extrinsics.



(a) F-BSPM (b) I-BSPM

Fig. 4.1: A comparative view on the forward and inverse mode of BSPMs. Left: The F-BSPM utilizes the measured proprioception together with behavior-specific knowledge to predict the intrinsics. The difference between expectation and measured proprioception is used to estimate the amount and direction of extrinsic perturbations. Right: In contrast, the I-BSPM estimates the behavior parameter which corresponds best to the measured proprioception. Therefore, extrinsic perturbations are defined by the difference between estimated and actually configured behavior parametrization.

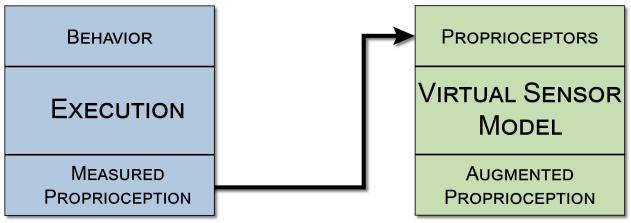


Fig. 4.2: The V-BSPM augments a robot's proprioception by combining available proprioceptive information. In contrast to the F-BSPM and I-BSPM, extracting the extrinsics is not addressed by the V-BSPM.

(2) The Behavior-Specific Proprioception Model in Inverse Mode (I-BSPM) is used to estimate behavior parameters that correspond best to the measured proprioception (rather than the expected proprioception). Here, the extrinsics is obtained by differentiating this estimate with the actual behavior configuration. Figure 4.1 gives a comparative depiction on the F-BSPM and I-BSPM modes. The main differences lie in their chronological sequence and the corresponding output spaces. Here, the F-BSPM predicts intrinsic values of a proprioceptor. For example, the expected measurements of an accelerometer or a force sensor. In contrast, the output of the I-BSPM is related to the behavior configuration. Hence, parameters which control the behavior are estimated, e.g., step length of a humanoid robot's walking behavior.

(3) Furthermore, this thesis proposes a V-BSPM. The corresponding model architecture is illustrated in Figure 4.2. Here, instead of detecting the extrinsics, a V-BSPM is applied to augment a robot's proprioception. More precisely, a V-BSPM estimates proprioceptive information by combining underlying correlations from the available proprioceptors. A V-BSPM therefore is applicable to augment the proprioception also in the absence of real world feedback, e.g., derive force information from joint currents.

One goal of the proposed BSPMs is to be applicable to arbitrary robot platforms without the need of expert knowledge, e.g., mass distribution or kinematic chains. In the following, a purely data-driven learning is proposed to extract sensory-motor experiences of the robot with regard to the particular behavior.

4.2 Learning BSPMs

The proposed BSPM approach is implemented by a purely data-driven training phase. First, a robotic behavior, which is defined by a sequence of motor commands, is executed under various conditions. Simultaneously, all available quantities such as (1) proprioceptors, (2) behavior parameters and (3) context state descriptions are recorded and stored as training examples. The acquired data therefore contains behavior-specific sensory-motor experiences, possible variances and correlations.

Among the available quantities, a *target* value is designated which is to be predicted from the other values by the BSPM. This target is directly related to the corresponding BSPM mode and can be chosen from any available quantity. In the following, one example for each kind of target is given and assigned to the particular BSPM mode:

1. F-BSPMs predict proprioceptors: Walking with a configured step length influences the center of mass of a humanoid robot.

2. I-BSPMs estimate behavior parameters: Joint configurations and motor torques contain information about the actual rather than the configured step length of a humanoid robot's walking gait.

3. V-BSPMs estimate context state: By monitoring the power consumption of its joints, a robot can estimate the weight attached to it.

As can be seen in these examples, targets are usually correlated to other parts of the proprioception. Therefore, the remaining quantities can be used as possible inputs to the BSPM while the target is defined as its output.

Not all of the recorded quantities are equally important for the execution of a specific behavior and therefore also contain different amounts of information about the particular target. For example, the step length of walking depends stronger on leg and hip joints than on fingers while this is vice versa for grasping behaviors. Therefore, the presented approach selects a subset of quantities, which are strongly correlated to the target, as inputs to the BSPM. In the following sections, the successive data acquisition, sensor selection and model learning steps are explained in more detail.

4.2.1 Data Acquisition

As stated above, the presented approach is purely data-driven which therefore requires the acquisition of training examples. To ensure the heterogeneity of these examples, the behavior needs to be executed under varying target configurations. However, depending on the particular type and application of the BSPM, this procedure differs significantly. More precisely, during data acquisition, the behavior may be executed under varying conditions:

1. Regular execution: the behavior is executed under controlled environmental conditions where no extrinsic perturbations affect it.

2. Perturbed execution [optional]: the robot's behavior is intentionally perturbed, e.g. by a human supervisor.

3. Augmentation with training-only sensor data or context descriptions [optional]: additional sensors[1] or contextual knowledge about the task environment may be available.

As a result, regular training examples contain only correlated information between the target and the intrinsics. This is vital when learning an F-BSPM or I-BSPM since it allows to derive the extrinsics from the measured proprioception.

In contrast, a V-BSPM does not differentiate between intrinsics and extrinsics. Hence, optional recordings from perturbed and augmented behavior executions can be added to the training examples. The contained intrinsic and extrinsic correlations enable the V-BSPM to estimate the overall proprioception rather than the intrinsics only.

For example, a V-BSPM can be used to approximate the functionality of a force sensor. Therefore, information about the forces occurring during regular and perturbed behavior executions need to be acquired. Later is achieved by manually applying forces to the robot during behavior execution. Learning the V-BSPM with these examples therefore allows to generalize forces during regular and perturbed behavior executions. Hence, the real force sensor can be replaced by its virtual counterpart. While maintaining accurate force estimates, this reduces costs and is beneficial for the robot's payload.

Furthermore, a V-BSPM can learn to assign context descriptions. For this, the user needs to enrich the recorded examples with contextual information, e.g., the user attaches a specific weight to the robot's end-effector. The corresponding context descriptions are stored as additional labels in the training examples and are used as target for learning. As a result, the V-BSPM automatically assigns the most reliable context description during runtime.

In order to keep the memory requirements of the proposed approach as low as possible only a small base of relevant training examples is acquired. More precisely, the applications presented in this thesis require one to three minutes of training data. The hardware used in these applications has frame rates within between 60 Hz and 125 Hz which further need to be synchronized by means of data preprocessing. Depending on complexity of the particular platform (number of joints and amount of sensors) one minute of training data may contain millions of values. These values represent the underlying sensory-motor dynamics of the behavior but also contain noise, redundancy and less relevant information. Furthermore, it is often unclear which of these sensory-motor readings need to be taken into account. Consequently, a preprocessing step which extracts the sensor readings with a high correlation to the target is vital to learn a well generalizing model from a small set of training examples.

[1] E.g. expensive, high-precision sensors that cannot be used in field operations due to logistical reasons

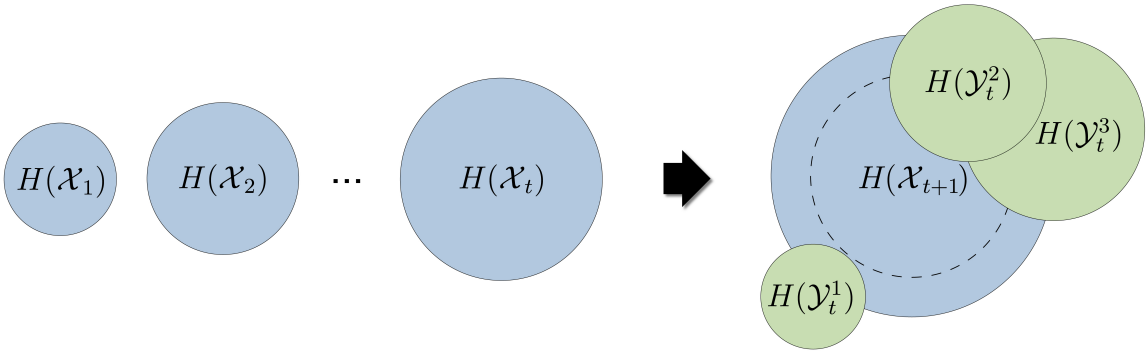


Fig. 4.3: The target process $\mathcal{X}$ (blue) with regard to simultaneous proprioception processes $\mathcal{Y}$ (green). The Shannon entropy of the target process $\mathcal{X}$ continuously increases over time. By utilizing its past states the target process can predict its future state $\mathcal{X}_{t+1}$. The target process is only certain about its own past (inside the dotted line) but is still uncertain about its future (outside the dotted line). Here, the proprioceptors $\{\mathcal{Y}_t^2, \mathcal{Y}_t^3\}$ share information with the state of the target $\mathcal{X}_t$ while $\{\mathcal{Y}_t^1, \mathcal{Y}_t^2, \mathcal{Y}_t^3\}$ contain information about the future of the target.

4.2.2 Sensor Selection

In the context of this thesis, correlated proprioceptors are automatically selected based on a data-driven approach or simply by a manual selection performed by the user. The latter is always a good choice when the user has a fair degree of knowledge about the platform, its environment and the behavior. However, with increasing complexity this also requires more and more user experience what therefore makes the approach less applicable. For instance, mutual interactions, indirect dependencies and delayed correlations are difficult to detect within millions of values. Reducing the dimensionality of the training data is a common solution to this problem and therefore is also utilized in several applications of this thesis. Here, dimensionality reduction methods reduce the training data while maintaining as much overall information as possible. However, the main drawback is that these methods do not examine correlations between the proprioception and the particular target. Consequently, a proprioceptor which contains important information about the target but less overall information may be erased from the data.

Therefore, a relation based method which automatically selects the most relevant proprioceptors is presented. More precisely, the following information-theoretic measures (see Section 3.1) are examined:

- Conditional Mutual Information (CMI)
- Transfer Entropy (TE)
- Multivariate Mutual Information (MMI)

These methods are based on the definition of Shannon entropy (see Equation 3.2 p. 27) and are only applicable for discrete sets of data. Therefore, all training examples need to be preprocessed. First, normalization is applied to unify the different data ranges of all sensor streams. Next, the continuous data space of these values is discretized

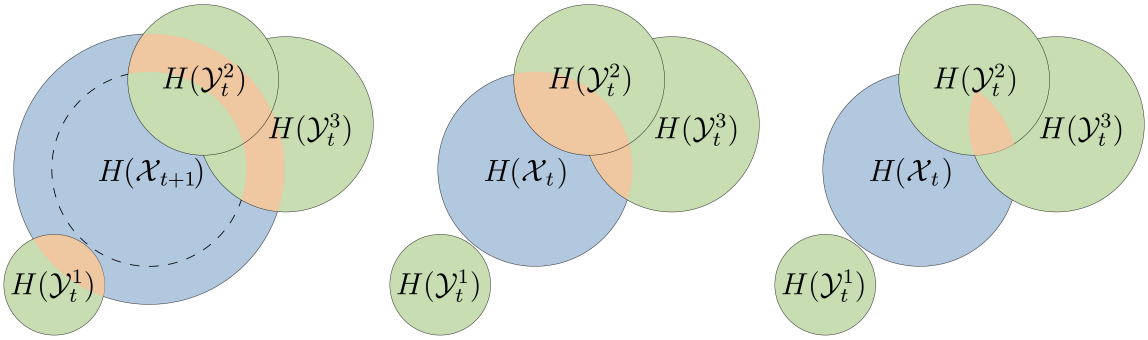


Fig. 4.4: Different information-theoretic measures between simultaneous processes. Left: TE (orange) measures the amount of information shared between $\{\mathcal{Y}_t^1, \mathcal{Y}_t^2, \mathcal{Y}_t^3\}$ and the future of $\mathcal{X}$. Middle: CMI (orange) measures the amount of information shared with the actual state of $\mathcal{X}$. Here, $\{\mathcal{Y}_t^2, \mathcal{Y}_t^3\}$ share information with $\mathcal{X}_t$ while $\mathcal{Y}_t^1$ is independent. Right: The redundant information shared between $\{\mathcal{Y}_t^2, \mathcal{Y}_t^3\}$ and $\mathcal{X}_t$. MMI (orange) allows to measure additional and already contained information and therefore determines redundant or synergistic effects.

to a fixed number of states. As a result, correlations between the target and the proprioceptors can be analyzed efficiently (see Section 2.2).

In the context of this thesis, a proprioceptor is considered as a process $\mathcal{Y}$ while the target is defined as process $\mathcal{X}$. Figure 4.3 illustrates a simple example where past information $H(\mathcal{X}_t)$ is beneficial to predict the next state $H(\mathcal{X}_{t+1})$. Here, the information inside the dotted line is already contained in the past of the target $\mathcal{X}_t$ while the area outside describes its uncertainty about the future.

The basic idea behind the usage of TE is to take into account the past of the proprioceptors $\{\mathcal{Y}_t^1, \mathcal{Y}_t^2, \mathcal{Y}_t^3\}$ to resolve this uncertainty. Figure 4.4 left highlights the areas where the target receives information about its own future from the proprioceptors $TE_{\{\mathcal{Y}^1,\mathcal{Y}^2,\mathcal{Y}^3\}\to\mathcal{X}}$ (see Equation 3.15 p. 34). As described in Section 3.1.4, TE can be written in terms of mutual information $I(\mathcal{X}_{t+1}; \{\mathcal{Y}_t^1, \mathcal{Y}_t^2, \mathcal{Y}_t^3\}|\mathcal{X}_t)$ (see Equation 3.18 p. 36). Therefore, TE determines the amount of proprioceptive information shared with the future of the target which is not contained in its own past. To improve the accuracy of target predictions, sensor selection prefers proprioceptors which transfer information to the future of the target. In contrast, $TE_{\mathcal{X}\to\{\mathcal{Y}^1,\mathcal{Y}^2,\mathcal{Y}^3\}}$ determines the information transfer between the target and the future of the proprioceptors. For example, proprioceptors which are strongly influenced by changes of the target can be useful to estimate its actual state. This non-symmetric property is utilized to apply an adequate sensor selection with regard to the particular task.

However, a proprioceptor with high TE does not necessarily contain simultaneous information. This fact is illustrated in Figure 4.4 middle. Here, $\mathcal{Y}_t^1$ is independent from the actual state of the target $\mathcal{X}_t$. Measuring such simultaneous effects is the core ability of mutual information. For example, the mutual information between the proprioceptor $\mathcal{Y}_t^1$ and the target is $I(\mathcal{X}_t; \mathcal{Y}_t^1) = 0$ (see Equation 3.7 p. 29). In contrast, $\{\mathcal{Y}_t^2, \mathcal{Y}_t^3\}$ have overlapping areas with the target and therefore $I(\mathcal{X}_t; (\mathcal{Y}_t^2, \mathcal{Y}_t^3)) \neq 0$. Furthermore, the detection of redundant proprioceptive information about the target can be beneficial for sensor selection. Hence, a closer look is taken on CMI and MMI.

One possibility is to extract the amount of shared information by utilizing CMI

(see Equation 3.9 p. 30). In particular, the information between the target and $\mathcal{Y}_t^2$ under the condition that $\mathcal{Y}_t^3$ is known is defined by $I(\mathcal{X}_t; \mathcal{Y}_t^2 | \mathcal{Y}_t^3)$. As depicted in Figure 4.4 middle, the information simultaneously shared between the target and the proprioceptors is determined by

$$I(\mathcal{X}_t; \mathcal{Y}_t^1) + I(\mathcal{X}_t; \mathcal{Y}_t^2 | \mathcal{Y}_t^1) + I(\mathcal{X}_t; \mathcal{Y}_t^3 | (\mathcal{Y}_t^1, \mathcal{Y}_t^2)).$$

As an alternative MMI quantifies the relation between the information growth and redundancy (see Equation 3.8 p. 30). Therefore, the redundant information depicted in Figure 4.4 right is determined by

$$I(\mathcal{X}_t; \mathcal{Y}_t^2; \mathcal{Y}_t^3) = I(\mathcal{X}_t; \mathcal{Y}_t^2) - I(\mathcal{X}_t; \mathcal{Y}_t^2 | \mathcal{Y}_t^3).$$

Here, a positive MMI indicates a redundant relation while negative values imply a growth of information or synergistic effect (see Equation 3.10 p. 31). Hence, both measurements are able to detect proprioceptors which contain shared information with the actual state of the target. This is appropriate for simultaneous effects but does not take into account delayed correlations. For this, the definitions of CMI and MMI are extended with a time delay δ

$$\mathrm{CMI}(\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \delta) = I(\mathcal{X}; \mathcal{Y} - \delta | \mathcal{Z}), \tag{4.1}$$

$$\mathrm{MMI}(\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \delta) = I(\mathcal{X}; \mathcal{Y} - \delta; \mathcal{Z}), \tag{4.2}$$

where $\delta = 0$ determines simultaneous influences. The resulting iterative sensor selection procedure with CMI is shown in Algorithm 1. Here, the proprioceptors $\mathcal{Y}$ are used to select the subset of sensors $\mathcal{Z}$ with maximal information about the target $\mathcal{X}$ where $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_n)$ determines n possible time delays. For this, the CMI between the target and the temporally shifted proprioceptors is computed. Since the condition $\mathcal{Z}$ is an empty set at the beginning, the first computation of CMI is equivalent to calculating the mutual information between the target $\mathcal{X}$ and the proprioceptors $\mathcal{Y}$. However, the

Algorithm 1 Sensor Selection utilizing Conditional Mutual Information

```
1: procedure SensorSelectionCMI(𝒳, 𝒴, δ)
2:     𝒵 ← {}
3:     while true do
4:         for all δ ∈ δ do
5:             cmi ← CMI(𝒳, 𝒴, 𝒵, δ)
6:         end for
7:         if max(cmi) > 0 then
8:             𝒵 ← 𝒵 ∪ 𝒴^argmax(cmi)
9:         else
10:            break
11:        end if
12:    end while
13: end procedure
```

proprioceptor which results in the highest amount of CMI is added to the condition $\mathcal{Z}$ by merging it with the already contained proprioceptors on line 8. The resulting condition is equivalent to the joint intersection of the particular sensor streams. To give a simple example, two sensors $\{[1,0,1,1,0];[1,1,1,0,0]\}$, which contain four state patterns $\{[1;1],[0;1],[1;0],[0;0]\}$, are merged to the condition $[1,2,1,3,4]$. By repeating this process, the information in $\mathcal{Z}$ about $\mathcal{X}$ is iteratively growing and therefore has a maximum of mutual information with the target. The procedure ends when no more additional information is contained in the remaining proprioceptors. The sequence of proprioceptors contained in $\mathcal{Z}$ therefore reflects the relevance about the target. A similar procedure is also applicable for the sensor selection with TE and MMI.

However, the reliability of these methods is usually restrained to the amount and quality of the training data. More precisely, a homogeneous set of training examples has a high probability to extract spurious correlations which are only correct for the space of training data but fail for runtime data. There are two possibilities to avoid this phenomenon. The first is to increase the user effort during data acquisition by regulating the amount and heterogeneity of the training examples. The second is to acquire an additional set of examples which are used for cross validation during learning.

4.2.3 Model Learning

The presented approach utilizes the sensory-motor experiences contained in the training examples to learn BSPMs. As stated above, the data acquisition phase intentionally kept short by recording only a small base of relevant examples. Therefore, in addition to accuracy, a major challenge of the learning procedure is to generalize target values for arbitrary proprioceptive measurements. To solve this, the model needs to extract the underlying sensory-motor dependencies of the robot with regard to the behavior. This process is aided by the previously discussed sensor selection approach but is also crucial for model learning. In the context of this thesis, different eager and lazy learning techniques (see Section 2.1) are evaluated empirically.

Lazy learning algorithms are based on a direct comparison between runtime and training data. A lazy learning technique presented in this thesis therefore makes use of Dynamic Time Warp (DTW). As introduced by Sakoe [106], DTW is a time series alignment algorithm for measuring the similarity between two temporal sequences. In this regard, the goal of the proposed approach is to find the optimal correspondence between training and currently observed sensations (see Figure 4.5 left). Here, a sequence of proprioceptive measurements is warped into the space of training data where the Euclidean Distance (EUD) determines their relative position. The target of the training example with the closest proximity is then used as an estimate of the actual one. Instead of a direct comparison, eager learning techniques generate an abstract model. After training, this model is used for predictions while the training examples are discarded. For the implementation of BSPMs, two eager learning approaches are utilized: Gaussian Process Regression (GPR) and Artificial Neural Networks (ANNs).

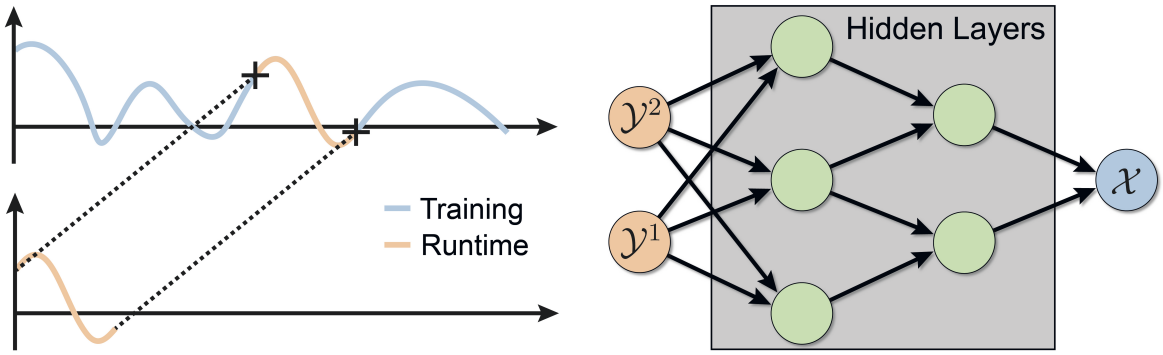


Fig. 4.5: Lazy learning and eager learning techniques are applied to generate BSPMs. Left: A lazy learner approach utilizes the DTW algorithm to align training and runtime data. The training data of the closest proximity is used as an estimate of the actual target. Right: ANNs are eager learning techniques which generate an abstract model by means of interconnected neurons. By iteratively adapting the connection weights with regard to the training data the network learns to generalize target values $\mathcal{X}$ for arbitrary proprioceptive inputs $\mathcal{Y}$.

GPR utilizes Gaussian Processs (GPs) to generate probabilistic models which specify a distribution over functions with one or more inputs. Therefore, they are feasible to perform a non-linear regression from some input space to an output space. In the context of this thesis, they are applied to map proprioceptive measurements to the most probable target values. As noted by Quiñonero-Candela and Rasmussen [107], one limitation is the cubic growth of computational complexity $\mathcal{O}(n^3)$. Hence, restraining the number of training examples is crucial for learning these models efficiently.

As extensively described in Section 3.2, ANNs have no such limitations. Here, a function is described by a network of interconnected neurons. The connection weights between neurons therefore determine the network's information processing. Neurons are further divided into an input, hidden and output layer. In the context of this, proprioceptors are used as input neurons while the target's dimensionality is assigned to the output neurons (see Figure 4.5 right). The hidden layers between input and output are also referred to as *black box* [108]. This is not because of 'hidden' or missing information about the network's configuration or functionality. In fact, an ANN is a fully defined deterministic function which approximates a more complex one. The term is utilized because the information flow is usually not relevant for the user and can be omitted. In particular, input and output are defined by the particular task while the number of hidden layers and neurons is potentially unrestricted and may greatly vary between concrete Learning is achieved by iteratively adapting the connection weights with regard to the particular inputs and outputs contained in the training examples. Here, the computational demand increases with the number of adaptable weights rather than the number of training examples. Hence, learning benefits from a large amount of training data and can be automatically stopped when a certain level of accuracy (see Section 3.2.3 p. 41) is reached. As a result, the network generalizes outputs for arbitrary inputs.

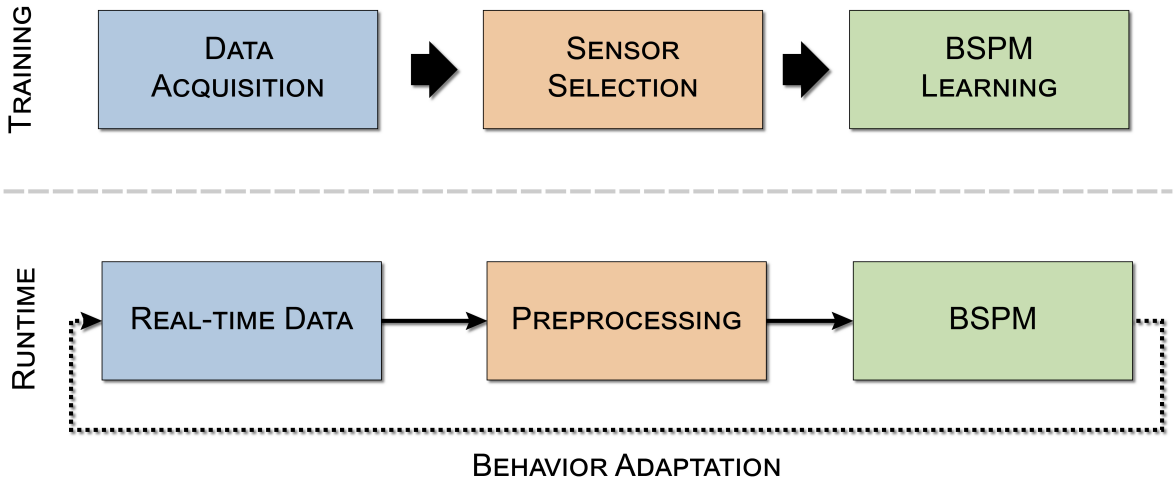


Fig. 4.6: An overview about the proposed BSPM approach. First, the behavior is executed while recording all available information during a training phase. Here, the most beneficial sensors are selected to increase the efficiency of BSPM learning. During runtime the data stream is preprocessed with regard to the selected sensors what therefore allows to apply the trained BSPM. Finally, reaction rules are implemented to adapt the behavior with regard to the predictions made by the BSPM.

4.3 Summary

The proposed machine learning approach is constructed as illustrated in Figure 4.6. First, a behavior is executed while continuously recording the proprioceptors together with the particular target. This is repeated for various target configurations to ensure the heterogeneity of the training examples. Depending on the particular type of model it is important to provide regular or perturbed executions of the behavior. More precisely, the F-BSPM and I-BSPM require training examples from regular behavior executions only. In contrast, a V-BSPM can also handle training examples which are recorded under perturbed circumstances. Next, the dimensionality of the proprioceptors is reduced by erasing redundant, noisy and spurious correlated data. Hence, sensors which are highly correlated to the future (TE) or the actual state of the target (CMI/MMI) are selected. This is beneficial for prediction purposes and ensures accurate and efficient model learning from few training examples. Here, the behavior-specific target parameter and the particular application determine which kind of model need to be applied.

F-BSPMs are utilized to predict intrinsic fluctuations where the target is part of the robot's proprioception. Therefore, the intrinsics of the target are estimated in the corresponding sensor space. For instance, when utilizing a force sensor the F-BSPM predicts the expected regular force. This prediction is compared to the actually measured force what therefore allows to estimate the amount and direction of extrinsic forces. The definition of adequate reaction rules further allows adapting the behavior with regard to the detected extrinsics. For example, the direction of the extrinsics can be used to swap the walking direction of a humanoid robot. Furthermore, an emergency stop could be implemented by comparing the amount of extrinsic forces with a predefined threshold. Hence, F-BSPMs are beneficial to implement applications

which require accurate sensing capabilities but require some user effort when adapting the behavior.

Similar to this, **I-BSPMs** reflect the intrinsics but estimate extrinsic perturbations in the behavior-specific space of possible actions. Hence, the target is an action or parameter which influences the execution of the behavior. For example, when performing a walking behavior the inverse model generates an estimate of the configured step length. The difference between this estimate and the actually configured step length is attributed to extrinsic forces which perturb the execution of the walking behavior. While the amount and direction of extrinsic perturbations are omitted, this is beneficial when implementing a control loop mechanism which automatically adapts the behavior to the given constraints. More precisely, no additional mapping from sensor readings to an adequate reaction needs to be provided by the user.

In contrast, **V-BSPMs** do not distinguish between intrinsics and extrinsics but augment the robot's proprioception by combining available proprioceptive information. This enables the robot to approximate the functionality of expensive, heavy or power-intensive devices with a V-BSPM. Furthermore, the user can manually assign context descriptions to the training examples. During runtime, the so far manually assigned descriptions are estimated by the V-BSPM and therefore augment the robot's proprioception. Similar to the F-BSPM, the augmented proprioception can be used to adapt the behavior by means of predefined reaction rules.

In the following, the proposed BSPM approach is applied to robots which require reliable and accurate force sensing capabilities. As discussed in Section 2.3 this, in general, is beneficial for the implementation of self-adapting behavior and a wide variety of tasks. In particular, this thesis focuses on robots which have to physically interact with their environment. Due to the inherent heterogeneity of robotics [109], it is important that an approach is applicable for different implementations and scales to various platforms and tasks. Therefore, the presented approach is independent from the size and sensor equipment of the robot and applies various model learning techniques. More precisely, the BSPM approach is applied to highly diverse robot platforms, e.g., a humanoid robot and an industrial robot arm which further have to solve very different tasks:

- The following chapter introduces F-BSPM and I-BSPM applications that infer guidance information during physical human-robot collaboration.
- Chapter 6 applies V-BSPMs to augment a robot's proprioception with accurate force sensing capabilities regarding tool usage and weight classification tasks.
- A combination of multiple BSPMs is utilized to accurately detect perturbing forces/torques during task execution in Chapter 7.

Also, the pros and cons of lazy and eager learning techniques as well as different BSPM modes (forward/inverse/virtual) will be discussed.

5 Inferring Guidance Information in Human-Robot Collaboration

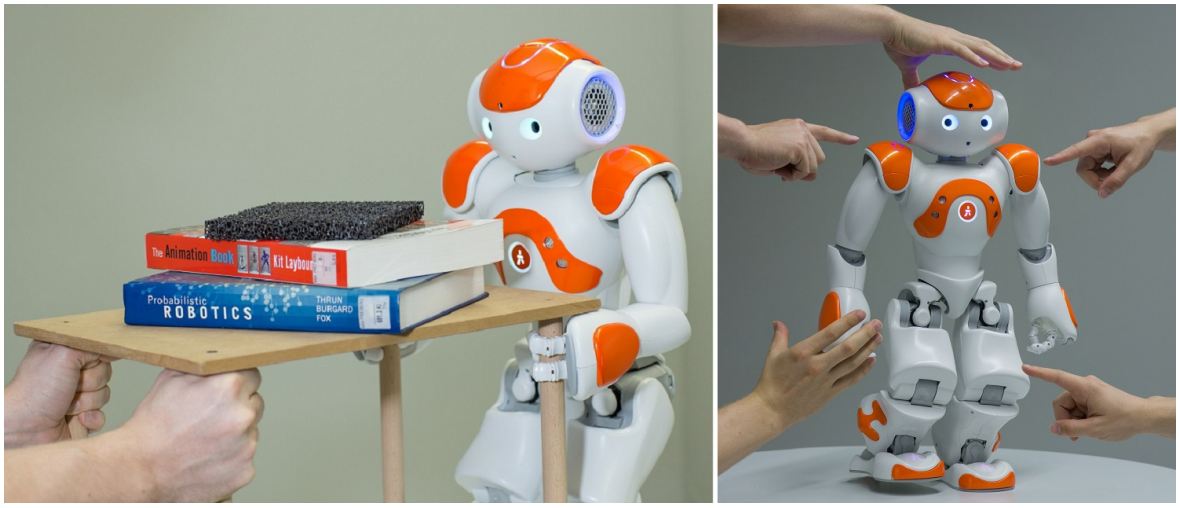


Fig. 5.1: A NAO robot is following the human guidance during a collaborative transportation task (left) or adapts its behavior to direct touch (right). The strength and direction of the corresponding extrinsic perturbations is measured by utilizing the proposed experience based BSPM approach.

In this chapter, the BSPM approach is applied to physical human-robot collaboration tasks where the goal is to give the robot similar capabilities as the human it replaces. Concretely, a transportation task and guidance through touch scenario are realized utilizing a NAO robot. The humanoid structure of this platform has the advantage that the human transfers his knowledge about human-human interactions to the human-robot setup. Hence, the human partner can apply his part of the interaction with some minor adaptations to the hardware constraints of the robot what ensures an intuitive interaction with the artificial assistant.

As shown in Figure 5.1, the human guides the robot, which continuously has to appropriately react to the exerted forces, e.g., walk forward or backward and perform side-steps. Here, it is assumed that the human forces cannot be directly measured using, for example, force-torque sensors but has to rely on the robot's standard proprioceptors instead. In particular, the NAO robot provides various proprioceptive measurements such as joint angles, positions, currents and foot pressure sensors. Furthermore, higher level *stability parameters* such as the center of pressure can be calculated from these measurements.

As proposed in Section 4.3, a BSPM is constructed in a process involving training data acquisition, correlation analysis and model learning. In the following sections,

two different BSPM implementations are presented:

1. An eager F-BSPM generates a statistical model to predict the most probable stability parameter intrinsics.

2. A lazy I-BSPM interpolates training examples to estimate the most similar behavior parameter configurations.

Here, the input to both BSPMs is derived from the measured proprioception which results from the execution of the particular behavior parameter configuration. Both implementations can be used to identify extrinsic perturbations in the proprioceptors, which are caused by the human. These extrinsics are then used to intuitively adapt the humanoid robot's behavior to the human guidance.

5.1 Eager F-BSPM Learning

The F-BSPM proposed in this section derives extrinsic forces from the prediction of higher-level stability parameters. To this end, the NAO robot provides access to the center of mass and further allows computing the center of pressure from its foot pressure sensors. Deviations between predicted and actually measured stability parameters are treated as indications for the guidance of the human, e.g., a push backwards signals the human's intention to move backwards.

For this, an F-BSPM is trained by regular proprioceptive measurements which are also referred to as intrinsics. This requires executing and recording the behavior under controlled environmental conditions where no extrinsic perturbations affect it. Utilizing the recorded data, a low-dimensional embedding is extracted by means of correlated information. The F-BSPM then learns a statistical model which maps low-dimensional projections to stability parameters for the particular behavior.

At runtime, the F-BSPM predicts the stability parameter intrinsics which occur due to the execution of the behavior. If the deviation between predicted and actually measured stability parameter value cannot be explained by the predictive uncertainty, the robot attributes this effect to an extrinsic perturbation caused by the human interaction partner. The sign of this deviation is then used to infer the human guidance and control the robot adequately. In the following, each step is explained in more detail.

5.1.1 Data Acquisition: Behavior Examples

In order to learn a predictive F-BSPM, representative data sets are collected for the relevant behaviors, e.g., knee bending and walking. The goal of this process is to gather intrinsic patterns of the proprioceptors and possible conditions to behavior parameters. Hence, behavior parameter specific intrinsics need to be recorded for a variety of configurations. For example, the NAO robot's walking behavior can be configured by a step length parameter which influences multiple joint angles. Hence, the training data need to include behavior executions for different step lengths. This

enables the F-BSPM to predict accurate intrinsics also for varying behavior parameter configurations.

The NAO robot provides various proprioceptors which include 24 joint angles, currents and positions as well as eight foot pressure sensors at a sampling rate of 100 Hz. To increase the robustness of the later model learning method, additional proprioceptors, so called higher-level stability parameters, are calculated. In particular, these are the robot's three-dimensional *center of mass* $\mathbf{c}^m = (c_x^m, c_y^m, c_z^m)$ and *center of pressure* $\mathbf{c}^p = (c_x^p, c_y^p, c_z^p)$ where c_x determines the longitudinal, c_y the lateral and c_z the vertical axis. The center of mass is calculated internally by utilizing the position $\mathbf{p} = (p_x, p_y, p_z)$ and mass m of each body part

$$\mathbf{c}^m = \frac{1}{\sum_{i=1}^{24} m_i} \sum_{i=1}^{24} \mathbf{p}(i) m_i. \tag{5.1}$$

In contrast to that, the robot's center of pressure is derived from the position $\mathbf{p}$ and measurement r of the foot pressure sensors

$$\mathbf{c}^p = \frac{1}{\sum_{i=1}^{8} r_i} \sum_{i=1}^{8} \mathbf{p}(i) r_i. \tag{5.2}$$

The main advantages of generating such higher-level stability parameters are their comprehensible manner, general usability for various behaviors and the reduced number of output values of the corresponding F-BSPM where the latter is also important for the model's input.

5.1.2 Feature Space

While it is often possible to utilize a large number of different proprioceptors these typically contain significant redundancy and noise. Additionally, it is often unclear which proprioceptor is most relevant to the particular behavior. Correlation analysis helps to single out important parts of the recorded training data.

It is generally known that human motion in principle is based on low-dimensional manifolds [50, 110, 111]. Hence, dimensionality reduction can be applied to recorded behaviors in order to remove redundant information. In particular, the recorded joint angles, which are the potential inputs for the predictive model, are reduced in dimensionality by utilizing manifold statistical procedures. Such procedures search for correlations between sets of joint angles.

The found correlations are then transformed into uncorrelated sets of variables, which are referred to as Principal Components (PCs). Contained in a so called *feature space*, the result is a set of low-dimensional points which represent the dynamics of the executed behavior. In the following, PCA [112], Locally Linear Embedding (LLE) [113] and Manifold Charting (MC) [114] are applied to a 60 second recording of the walking behavior. Figure 5.2 left shows the first PC of these procedures for the NAO robot's 24 joint angles. Here, the behavior parameter of the step length and therefore the walking

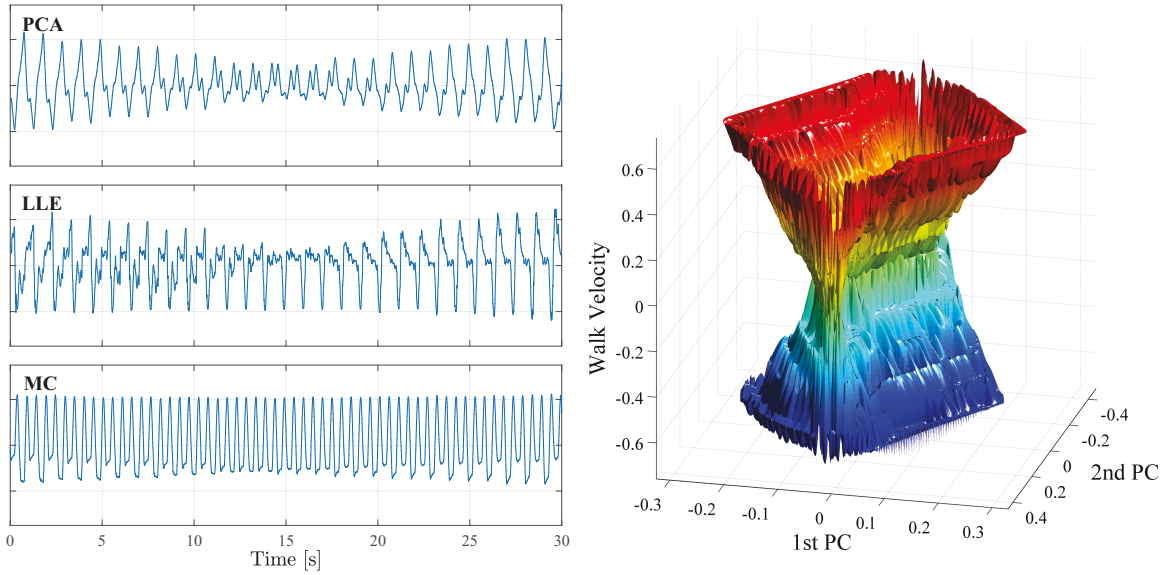


Fig. 5.2: The joint angle dimensionality of a walking behavior is reduced to a subset of PCs. Left: The PC with the highest information content for various dimensionality reduction methods. All methods extracted a clear periodic behavior from the walking behavior while PCA best matches the proportion to the walking step length. Right: Resulting from the robot's 24 joint angles, the two-dimensional PCA feature space of the walking behavior with regard to its corresponding normalized velocity. Here, positive velocities are highlighted red while negative ones are marked blue. These two PCs still contain more than 95% of the overall information content and therefore are utilized for learning the F-BSPM.

velocity is gradually decreased beginning from the maximum forward velocity to the maximum backward velocity. As can be seen for all methods, the executed motor behavior exhibits a clear periodic behavior what will be exploited during the later model learning process. Furthermore, for the usage of PCA, the amplitude of the first PC best matches the proportion to the robot's walking velocity while a less salient effect can be observed when utilizing LLE and MC.

This corresponds to the findings of Ben Amor [50] who intensively discusses the pros and cons of various dimensionality reduction methods for humanoid motions. Also, PCA does not require any kind of hyper-parameters and further achieves a better tradeoff between computational demands and quality of results. Here, the PCs are extracted using an eigenvector decomposition of the sensor data matrix. The eigenvalues define how much variance each eigenvector carries and therefore the highest eigenvalue is the direction with highest variance. Hence, in traditional PCA the relevance of a sensor stream is defined by the observed variance along that dimension.

However, after utilizing PCA, the 24 joint angles are replaced by the PC trajectories with the highest eigenvalue. Figure 5.2 right shows the resulting two-dimensional feature space with regard to the normalized walking velocity parameter. These two PCs contain more than 95% of the variance in the recorded training data. This has the benefit of significantly reducing the complexity of the learning task, while at the same time dampening the effect of noise and outliers, as these are often cut out from these PCs.

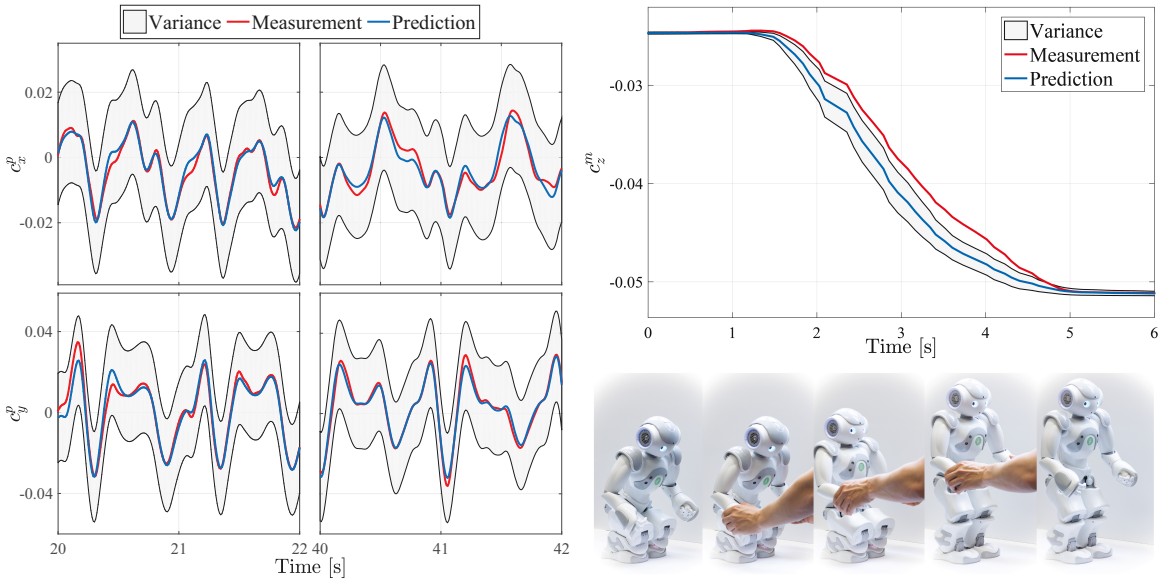


Fig. 5.3: The F-BSPM predicts stability parameter intrinsics. Left: The F-BSPM predicts the longitudinal c_x^p and lateral position c_y^p of the center of pressure. The deviation between measured values and the learned prediction is located inside the allowed variance which is given by the predictive uncertainty of the Gaussian process. Right: The robot stands up while inferring that the human is pulling at its arms. The execution of the corresponding knee bending behavior is driven by the difference in the predicted and measured vertical center of mass c_z^m.

5.1.3 Learning a Probabilistic Model

Subsequently, the acquired training data is used to learn the F-BSPM. For this, the PCA features and the actual behavior parameter configuration are used as input while the F-BSPM predicts stability parameters such as the center of mass. Here, Gaussian Process Regression (GPR) [115], a widely used method for generating a probabilistic, non-linear mapping between two data sets, is applied. Figure 5.3 left compares learned and measured pressure values for longitudinal walking while no extrinsic perturbations occur. As can be seen, only small deviations between predicted (blue trajectory) and measured pressure (red trajectory) are detected. These deviations are caused by the predictive uncertainty of the GPR model. One advantage of GPR is the ability to calculate the variance (highlighted gray) for any predicted value of the behavior. This allows to distinguish between model induced variability and extrinsic perturbations which are detected when the prediction exceed the variance.

In more detail, a Gaussian Process (GP) models a distribution $p(f)$ over functions, where f is a function mapping some input space to an output space. Given a set of training data $(\mathbf{X}, \mathbf{y}) = ((\mathbf{x}(i), y_i)|i = 1, \ldots, n)$, containing n samples of proprioceptive input values $\mathbf{x}$ resulting in a stability parameter output y, a non-linear regression can be written as

$$y = f(\mathbf{x}) + \epsilon \sim \mathcal{N}(0, \sigma^2). \tag{5.3}$$

Here, it is assumed that the variability follows a Gaussian distribution with zero mean

and variance σ^2. The prior of f is a GP

$$p(f) = \mathcal{GP}(\mu, \mathbf{K}), \tag{5.4}$$

which is specified by its mean μ and covariance matrix $\mathbf{K}$. The training data $(\mathbf{X}, \mathbf{y})$ is used to adjust these parameters by performing Bayesian regression

$$p(f|(\mathbf{X}, \mathbf{y})) = \frac{p(f)p((\mathbf{X}, \mathbf{y})|f)}{p(\mathbf{X}, \mathbf{y})}. \tag{5.5}$$

The resulting parameter configurations of the GP specify the posterior of f. Hence, the stability parameter prediction $\hat{y} \sim \mathcal{N}(\mu(\hat{\mathbf{x}}), \sigma^2(\hat{\mathbf{x}}))$ for a new proprioceptive input $\hat{\mathbf{x}}$ has Gaussian probability distribution with mean $\mu(\hat{\mathbf{x}})$ and variance $\sigma^2(\hat{\mathbf{x}})$

$$\begin{aligned} \mu(\hat{\mathbf{x}}) &= \mathbf{K}(\hat{\mathbf{x}}, \mathbf{X})[\mathbf{K} + \sigma^2\mathbf{I}]^{-1}\mathbf{y} \\ \sigma^2(\hat{\mathbf{x}}) &= \mathbf{K}(\hat{\mathbf{x}}, \hat{\mathbf{x}}) - \mathbf{K}(\hat{\mathbf{x}}, \mathbf{X})[\mathbf{K} + \sigma^2 I]^{-1}\mathbf{K}(\mathbf{X}, \hat{\mathbf{x}}). \end{aligned} \tag{5.6}$$

Since offsets and simple trends can be subtracted out from data, e.g., by performing PCA, a zero mean prior of the GP can be used. Thus, the key quantity is the covariance matrix $\mathbf{K}$, whose elements are the output of the covariance function or kernel. In the context of this thesis, the squared exponential kernel is utilized. For a detailed mathematical derivation of GPR the interested reader is referred to [115, p. 7-31].

5.1.4 Stability Parameter Estimation

During human-robot interaction the actual joint angles of the robot are measured and projected into the feature space. The resulting low-dimensional features and the actually configured behavior parameter, e.g., the walking step length, are mapped to the desired stability parameter via the F-BSPM. This predicted reference value represents the intrinsics which occur due to actual execution of the behavior. An extrinsic perturbation is assumed if the measurement is located outside the predictive variance of the Gaussian process.

Here, an extrinsic perturbation is perceived by the human interaction partner who is guiding the robot. The robot has to react to the strength and direction of this perturbation by continuously adapting its behavior parameters or changing its pose. For instance, the robot decreases the walking velocity when the human pushes it or the robot stands up while pulling at its arms. More precisely, a feedback loop, based on the difference between the predicted intrinsic and the measured stability parameters is utilized to adapt the behavior. As a result, the robot reacts to the human guidance.

In the following section, the proposed F-BSPM implementation is evaluated for the introduced cooperative transportation task.

5.1.5 Evaluation

Two different tasks are taken into account to illustrate the functionality of this eager F-BSPM implementation: a collaborative transportation task and guidance through touch scenario. The former enables both interactants to carry an object while the latter allows direct manipulation of the robotic behavior. Hence, the human guides the robot in both scenarios by direct (touch) or indirect (transportation) forces.

In order to realize the desired interaction capabilities different F-BSPMs need to be learned. In particular, the transportation task and the direct guidance scenario are both realized by the following behaviors:

1. knee bending
2. longitudinal walking
3. lateral walking

Behavior (1) allows the robot to move from a kneeling to an upright pose and vice versa. In contrast to that, behavior (2) enables the robot to walk forward/backward while behavior (3) is implementing the left/right walking direction. For (2) and (3) the behavior is adapted by a step length parameter which is influencing the corresponding walking velocity. While (2) and (3) can also be realized using a multivariate output GP, a better accuracy is experienced when using separate models.

A video presenting the overall results of the conducted experiments in this section can be found under the following link: `https://youtu.be/48y0hEix2fY`.

Knee Bending F-BSPM

In contrast to the two walking behaviors, knee bending is only a sequence of motor commands and is not configured by a behavior parameter. This simplifies the proposed model learning approach. More precisely, the robot moves from a kneeling to an upright pose and vice versa three times in a loop. During this, its 24 joint angles and the proposed stability parameters are recorded resulting in 2000 proprioceptive samples. Next, the dimensionality of the joint angles is reduced to the first PC (the one with the highest eigenvalue) by utilizing PCA. The resulting one-dimensional trajectory is used as input to the predictive F-BSPM. For measuring the strength and direction of the human guidance during knee bending, e.g., lifting an object, the vertical position of the center of mass c_z^m is most suitable. Hence, it is manually selected as the target of the F-BSPM.

At runtime, the learned model is used to predict c_z^m from the corresponding low-dimensional projection of the actual joint angle configuration. Figure 5.3 right illustrates the measured c_z^m (red) and predicted center of mass $\hat{c_z^m}$ (blue) when utilizing this model during pulling the robot by direct touch. Here, the robot is controlled by the manually defined reaction rules below:

- $c_z^m > \hat{c_z^m} + \sigma^2$: the robot moves to the next pose of the knee bending motion.

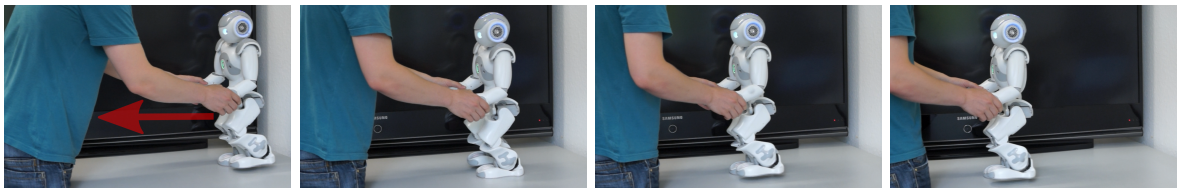


Fig. 5.4: The human directly guides the robot through longitudinal physical interaction. An F-BSPM is utilized to predict the intrinsic longitudinal center of pressure. Deviations between measured and predicted values are then utilized to infer the intended walking direction. Accordingly, the robot adapts the longitudinal step length parameter of the corresponding walking behavior.

- $(c_z^m \leq \hat{c_z^m} + \sigma^2) \wedge (c_z^m \geq \hat{c_z^m} - \sigma^2)$: the robot does not change its pose.
- $c_z^m < \hat{c_z^m} - \sigma^2$: the robot moves to the previous pose of the knee bending motion.

These rules can also be extended to the lifting and placing of objects by additionally taken into account a fixed offset of the object's weight. To further transport an object the direction and velocity of the robot's walking behavior need to be adapted. For this, predictive models of the lateral and longitudinal walking behavior are generated as follows.

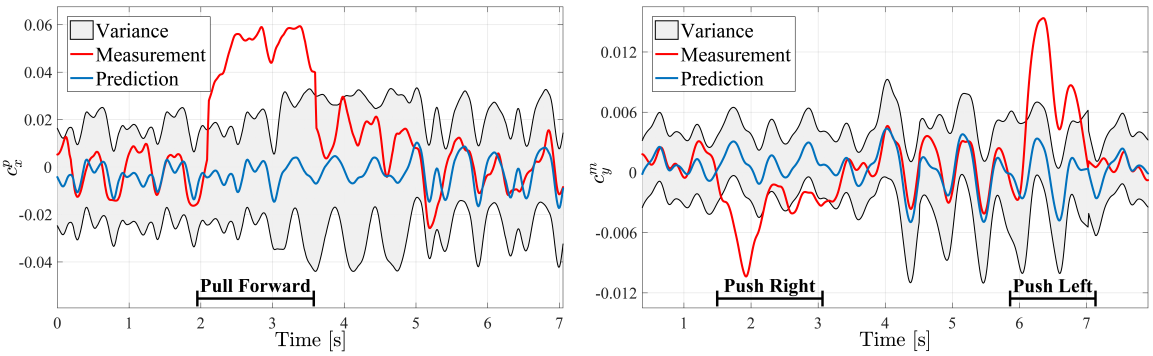
Longitudinal Walking F-BSPM

In contrast to the knee bending motion, the walking behavior is configured by a step length parameter which adapts the robot's walking velocity and direction. To find dependencies between such behavior parameters and the joint angles of the robot, they need to be adapted and recorded during data acquisition. Here, the robot starts with the maximum forward walking velocity and, during one minute, decreases its speed until walking backward with the maximum velocity. While doing so, the robot's joint angles, stability parameters and the step length parameter are recorded resulting in an overall of 6000 samples.

In order to learn an efficient predictive model the PCA is utilized. For the recorded data, two PCs which contain more than 95% of the variance are calculated. Together with the actually configured behavior parameter, these two dimensions are utilized as input to the F-BSPM. Furthermore, the longitudinal center of pressure c_x^p is selected as the model's output.

In this experiment, direct physical interaction between a human and robot through touch, as shown in Figure 5.4, is realized. More precisely, the human guides the robot by applying forces to its hands which are influencing the pressure measured at its feet. For this, the low-dimensional projection of the proprioceptors together with the actually configured step length is used to predict the center of pressure via the F-BSPM. Figure 5.5(a) shows the measured c_x^p (red) and predicted longitudinal center of pressure $\hat{c_x^p}$ (blue) when utilizing this model during direct physical interaction. In order to respond to these forces, the step length of the walking behavior is controlled as described below:

- $c_x^p > \hat{c_x^p} + \sigma^2$: the robot increases the longitudinal step length.



(a) Longitudinal Walking (b) Lateral Walking

Fig. 5.5: The difference between measured and F-BSPM predicted intrinsic stability parameter indicate extrinsic perturbations during human-robot interaction. Left: The longitudinal walking behavior is adapted by comparing the measured and predicted longitudinal center of pressure. The marked region indicates the time period where the human directly pulls the robot. Consequently, the robot increases its longitudinal walking step length. Right: The lateral walking behavior is adapted by the difference between measured and predicted lateral center of mass during performing a cooperative transportation task. The prediction differs from the measured value whenever the robot is pushed to the left or to the right. The robot responds by adapting the lateral step length accordingly.

- $(c_x^p \leq \hat{c}_x^p + \sigma^2) \wedge c_x^p \geq \hat{c}_x^p - \sigma^2)$: the robot does not change the longitudinal step length.
- $c_x^p < \hat{c}_x^p - \sigma^2$: the robot decrease the longitudinal step length.

These intuitive rules are also applicable for the transportation task. For this, the weight of the transported object and the corresponding forces are evenly distributed on the robot's feet. Hence, the carried mass is influencing the amount but not the position of the pressure and therefore the learned F-BSPM is independent from the weight.

Up to second two the velocity remains constant because the measurement is located inside the envelope. Between second two and four the measured center of pressure is outside of the allowed deviation, which triggers an increase of the step length and consequently a higher walking velocity. As a result, the robot follows the human guidance. It can be observed, that the prediction is adapted to the new walking velocity after a short delay. Since the human permanently interacts with the robot, also small variations are measured throughout the entire interaction.

Lateral Walking F-BSPM

Next, as shown in Figure 5.6, the proposed approach is applied to lateral walking in which the human applies forces to the robot via a carried object. Similar to the longitudinal walking model, also a one minute recording is utilized for different side stepping parameter configurations. The first two PCs, together with the step length, are utilized as input for the F-BSPM. In contrast to longitudinal walking, the lateral center of mass c_y^m is most suitable to detect lateral perturbations. This is related to the observation that the robot's hands and arms are not completely stiff. Hence, any forces of the human induce slight changes to their position and influence the c_y^m.

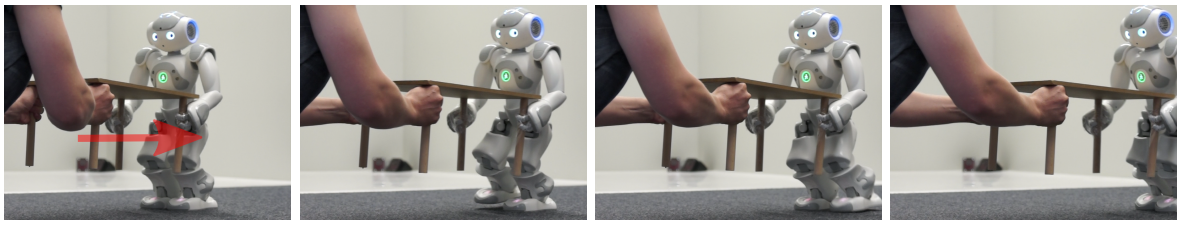


Fig. 5.6: The human applies lateral forces on the robot via a carried object during performing a cooperative human-robot transportation task. The robot utilizes an F-BSPM to predict the intrinsics of the lateral center of mass. These predictions are compared with the actually measured values where deviations are compensated by adequate adaptation of the lateral step length parameter.

By utilizing reaction rules similar to the ones for longitudinal walking the robot adapts its lateral step length parameter accordingly:

- $c_y^m > \hat{c_y^m} + \sigma^2$: the robot increases the lateral step length.
- $(c_y^m \leq \hat{c_y^m} + \sigma^2) \wedge (c_y^m \geq \hat{c_y^m} - \sigma^2)$: the robot does not change the lateral step length.
- $c_y^m < \hat{c_y^m} - \sigma^2$: the robot decrease the lateral step length.

The resulting measured c_y^m (red) and predicted lateral center of mass $\hat{c_y^m}$ (blue) are shown in Figure 5.5(b). Here, the human first guides the robot to the right and then to the left where an adaptation of the step length can be seen by the changing predictive variance.

5.1.6 Conclusion

In this section, the F-BSPM approach was applied to collaborative human-robot interaction tasks where an overview can be seen in Figure 5.7. First, training examples are gathered from a few regular executions of the particular behavior. For example, the walking behavior F-BSPM was learned with only one minute of training data. This ensures that training can be applied with a minimum effort and without the need of an additional simulated environment.

Next, the dimensionality of the recorded proprioceptors is reduced to a low-dimensional feature space by utilizing PCA. The trajectories along the most relevant PCs and the corresponding behavior parameter configuration are then used to learn an abstract model. More precisely, GPR was utilized to learn F-BSPMs of a humanoid robot's knee bending and two walking behaviors. These intrinsic proprioception models are then used to predict intrinsic stability parameters from the projection of the measured proprioception during behavior execution. Deviations between the predicted and measured stability pattern are attributed to the influence of human guidance. Finally, intuitive reaction rules need to be defined for an adequate behavior adaptation. It is shown that the F-BSPM approach in general can be used to realize collaborative tasks involving a human and a robot.

A drawback of the proposed implementation is the amount of knowledge required for the calculation and selection of the introduced stability parameters. For instance, the

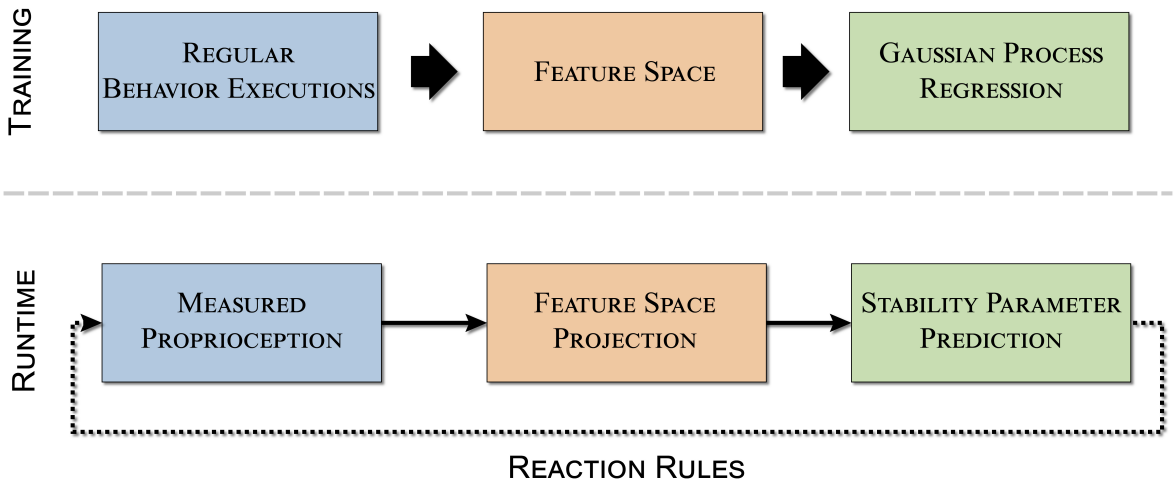


Fig. 5.7: Overview of the presented eager F-BSPM with regard to the methodology introduced in Section 4.3. During training, proprioceptors and stability parameters are recorded during regular behavior executions. Next, PCA is applied to the recorded training data resulting in a low-dimensional feature space. The poses contained in this feature space are then used to learn a mapping to the corresponding stability parameter configuration by utilizing GPR. At runtime, the measured proprioception is projected into the feature space. The corresponding low-dimensional pose allows predicting the most probable stability parameter. The difference between predicted and actually measured stability parameter is used to adapt the behavior by utilizing intuitive reaction rules.

robot's center of mass requires detailed information about the mass distribution and kinematic structure of the particular platform. This makes the approach less applicable for non-experts.

A further drawback concerns in the difference of the measurement spaces. More precisely, the intrinsics are predicted for the corresponding stability parameter while the behavior is controlled by a parameter configuration such as the step length of the walking behavior. Hence, the human needs to define task-specific interaction rules what reduces the overall applicability of the approach. This can be addressed by utilizing an I-BSPM since it directly estimates behavior parameter configurations.

Furthermore, an important advantage of utilizing GPR as core component of the presented approach, is its ability to learn a probabilistic model from a small set of training examples. However, as noted by Quiñonero-Candela and Rasmussen [107], when processing large data sets, training a GP often becomes computationally intensive. This is due to its cubic growth of computational complexity $\mathcal{O}(n^3)$. Various solutions have been proposed in the literature to cope with these situations such as pseudo-inputs [116] and sample selection [117] with moderate success. Hence, restraining the number of training examples is crucial for learning these eager F-BSPMs efficiently.

In the following, the benefit of utilizing a lazy learning technique which does not rely on an abstract model is introduced.

5.2 Lazy I-BSPM Learning

In order to reduce the computational demand for large data sets, the previously introduced eager approach is replaced by a lazy learning technique. The lazy structure of the algorithm therefore does not require computing an abstract model. Hence, a compact description of raw sensor readings into stability parameters is not necessary anymore. Furthermore, an I-BSPM is utilized that estimates perturbations directly in the corresponding behavior parameter space. This allows performing an adequate behavior adaptation without the need of predefined reaction rules.

In particular, it will be shown that a lazy I-BSPM algorithm can be applied to the introduced human-robot interaction tasks. Here, proprioceptors include only raw sensor readings such as joint angles and pressure sensors which are utilized to generate a low-dimensional space. To generalize outputs for arbitrary inputs additional examples are generated from a small set of original training examples. For this, a novel interpolation method from the field of fluid dynamics is utilized. Furthermore, information theory is applied in order to weight each PC by its relevance on the behavior parameter. The resulting scaled low-dimensional space is used to compare the actual proprioception with the proprioception perceived during training. The parameter configuration which results in the most similar proprioception is then used as an estimate of the actual behavior parameter.

5.2.1 Data Acquisition: Behavior Examples

One goal of the proposed BSPM approach is to be applicable without the need of expert knowledge about the particular robot platform. E.g., utilizing information about a robot's mass distribution or kinematic structure delimits the applicability of the proposed approach. The introduced stability parameters of the above F-BSPM make use of such information and therefore are excluded from the training data. Instead, only the 24 joint angles and the eight foot pressure sensors will be utilized for I-BSPM learning.

As stated in Section 2.1, lazy learning techniques do not generate an abstract model. Instead, generalization needs to be addressed by a wide variety of training examples which requires recording a behavior under a large set of possible parameter configurations. Since the parameter space may have a dynamic range, this can lead to a time-consuming recording phase, which in consequence leads to wear and tear of the robot hardware. To avoid a lengthy training session, a novel interpolation method is applied to only a few training examples.

Similar to the proposed F-BSPMs, regular behavior executions are recorded for an equidistant set of behavior parameters. For example, different step lengths of the NAO robot's walking behavior. In particular, recording a behavior for one parameter

configuration y results in

$$\mathbf{X}_1 = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}, \mathbf{y}_1 = \begin{pmatrix} y_1 = y \\ \vdots \\ y_n = y \end{pmatrix}, \tag{5.7}$$

where n is the number of samples and m is the number of proprioceptors. Hence, $\mathbf{X}_1$ is composed of n row vectors $\mathbf{x} = (x_1, \ldots, x_m)$ which contain m proprioceptive measurements while $\mathbf{y}_1$ contains the corresponding behavior parameter configuration. The training data for k equidistant parameter configurations then comprises a total of $n \cdot k$ samples

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_k \end{bmatrix}, \mathbf{y} = \begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_k \end{pmatrix}. \tag{5.8}$$

Since the behavior was executed under regular environmental conditions (without extrinsic perturbations) the training data contains correlations between the behavior's intrinsics $\mathbf{X}$ and the behavior parameter configuration $\mathbf{y}$. As will be shown later, these correlations are utilized to strengthen the mapping between training and runtime measurements. Furthermore, the exact equidistance of recorded parameter configurations is vital for the interpolation scheme which is introduced in the following.

5.2.2 Interpolation

The proposed lazy I-BSPMs do not learn an abstract model to generalize outputs for arbitrary inputs. Instead, training examples and runtime data are compared where the closest match is used to estimate the actual behavior parameter. The accuracy of this procedure strongly depends on the number of training examples where a lengthy training session would decrease the applicability and further increase the wear and tear of the robot's hardware.

An efficient way to expand the amount of training data can be achieved by using interpolation schemes. To this end, a novel interpolation method from fluid dynamics, so called Dynamic Mode Decomposition (DMD) [118][38] is applied. This method presents a modal decomposition for non-linear flows and allows the extraction of coherent structures oscillating at a single frequency and growth/decay rate. The basic idea is that DMD computes a linear model which approximates the underlying non-linear dynamics of an unknown system. Similar to the periodic nature of flows, it is assumed that DMD is applicable to the proprioception of periodic robotic behaviors, e.g., walking. Hence, DMD is used to obtain the behavior specific dynamics which can be used to interpolate proprioceptive measurements for arbitrary behavior parameter configurations.

Here, a recording for one behavior parameter is reconstructed by $\mathbf{z} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)^\mathsf{T}$ where m is the number of proprioceptors and $\mathbf{x} = (x_1, \ldots, x_n)$ contains n samples. The overall training data for k equidistant behavior parameter configurations is given by

$\mathbf{Z} = [\mathbf{z}_0, \ldots, \mathbf{z}_{k-1}]$. Hence, each column of $\mathbf{Z}$ contains the stacked temporal evolution of all proprioceptors for one of the equidistant behavior parameter configurations. This data is divided into two matrices $\mathbf{Z}_1 = [\mathbf{z}_0, \ldots, \mathbf{z}_{k-2}]$ and $\mathbf{Z}_2 = [\mathbf{z}_1, \ldots, \mathbf{z}_{k-1}]$. Consequently, these matrices are shifted by one sample and can be linked via a system matrix. Since the data is obtained from experiments, this system matrix is unknown and needs to be calculated for a very large system. As noted by Bagheri [119] it is computationally impossible to solve the eigenvalue problem directly as well as to fulfill the storage demand. Instead, the idea is to approximate the eigenvalue problem such as proposed by Chen, Tu, and Rowley [120] who project the system matrix onto a Krylow subspace.

In contrast, Schmid [118] describes a more robust calculation referred to as General Dynamic Mode Decomposition (GDMD). The GDMD is based on the linearization of the last snapshot, which is assumed to correspond to the underlying non-linear dynamics. This is achieved by applying a singular value decomposition on $\mathbf{Z}_1$ such that $\mathbf{Z}_1 = \mathbf{U}\boldsymbol{\Sigma}\mathbf{W}^*$. The *full-rank* matrix $\mathbf{S}$ is determined on the subspace spanned by the orthogonal basis vectors $\mathbf{U}$ of $\mathbf{Z}_1$, described by $\mathbf{S} = \mathbf{U}^*\mathbf{Z}_2\mathbf{W}\boldsymbol{\Sigma}^{-1}$. Solving the eigenvalue problem $\mathbf{S}\boldsymbol{\mu} = \boldsymbol{\lambda}\boldsymbol{\mu}$ leads to a subset of complex eigenvectors $\boldsymbol{\mu}$. The GDMD modes are then defined by $\boldsymbol{\Phi} = \mathbf{U}\boldsymbol{\mu}$ where the complex eigenvalues $\boldsymbol{\lambda}$ contain growth/decay rates and frequencies of the corresponding modes. The discrete temporal evolution of the GDMD modes is determined by the Vandermonde matrix

$$\mathbf{V}_{and} = \begin{bmatrix} 1 & \lambda_1 & \cdots & \lambda_1^{k-2} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_{k-1} & \cdots & \lambda_{k-1}^{k-2} \end{bmatrix}. \tag{5.9}$$

In order to perform an interpolation, $\boldsymbol{\Phi}$ is scaled by $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{k-1})$. The analysis of $\mathbf{V}_{and}$ shows that the first sample $\mathbf{z}_0$ is independent of the temporal evolution. Hence, the scaling factors $\boldsymbol{\alpha}$ can be calculated by solving $\boldsymbol{\Phi}\mathbf{D}_\alpha = \mathbf{z}_0$, where

$$\mathbf{D}_\alpha = \begin{bmatrix} \alpha_1 & & 0 \\ & \ddots & \\ 0 & & \alpha_{k-1} \end{bmatrix}. \tag{5.10}$$

Finally, the training data is reconstructed by $\mathbf{Z}_1 = \boldsymbol{\Phi}\mathbf{D}_\alpha\mathbf{V}_{and}$ and an arbitrary accurate interpolation is achieved by adding additional $\boldsymbol{\lambda}$ to $\mathbf{V}_{and}$. In particular, adding $\mathbf{v}_{and} = (\lambda_1^{1.5}, \ldots, \lambda_{k-1}^{1.5})^\mathsf{T}$ interpolates a new sequence of samples within between the second and third measurement of $\mathbf{Z}$.

The GDMD algorithm provides a rank-$(k-1)$ solution, which means that the approximate eigenvalues agree with the number of snapshots in $\mathbf{Z}_1$. By adding additional snapshots which do not increase the vector space (they do not contribute any new information to the system) the number of approximate eigenvalues still increases. Hence, the key challenge is to identify a subset of modes which capture the most important dynamic structures in order to achieve a good quality approximation. To solve this, the Sparsity-Promoting Dynamic Mode Decomposition (SDMD) was developed by Jo-

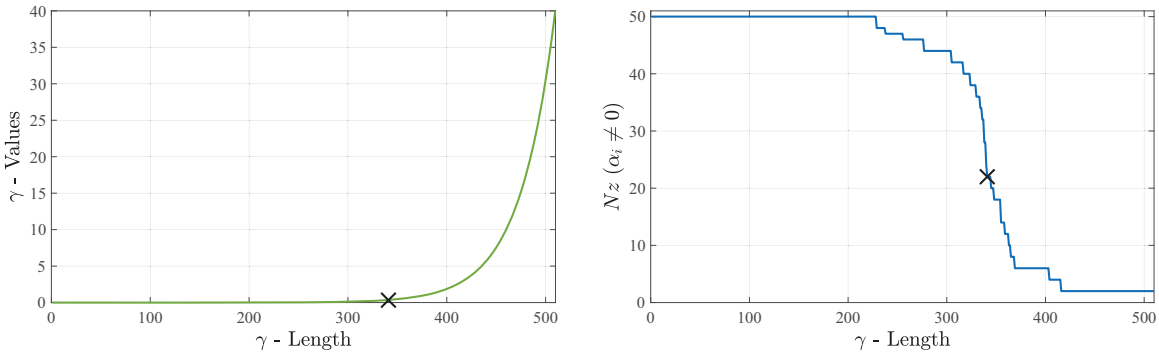


Fig. 5.8: The SDMD regularization parameter γ and its influence on non-zero amplitudes where the black cross marks $\gamma = 0.36$. Left: The curve progression of the regularization parameter γ. Right: The number of non-zero amplitudes (Nz) as function of γ. Here, $\gamma = 0.36$ results in 22 non-zero amplitudes. Here, a low number of non-zero amplitudes in $\boldsymbol{\alpha}$ helps to reduce the influence of disturbances in the approximation.

vanovi, Schmid, and Nichols [121]. Instead of scaling the whole spectrum of available modes, the SDMD only concentrates on the most dominant modes for the entire series by setting the amplitudes of the negligible modes to zero. For this, the problem is brought into the following constellation

$$\min_{\boldsymbol{\alpha}} \mathbf{J}(\boldsymbol{\alpha}) + \gamma \sum_{i=1}^{k-1} |\alpha_i| \,, \tag{5.11}$$

where γ denotes a regularization parameter that indicates the focus on sparsity of $\boldsymbol{\alpha}$. Larger values of γ increase the focus on sparsity (less extracted modes) as illustrated in Figure 5.8 for an exemplary robot data set with 51 examples. The higher the value of γ (green curve), the lower the number of non-zero amplitudes Nz (blue curve) and the more SDMD concentrates on the low-frequency modes. For the example presented here, $\gamma = 0.36$ which results in $Nz = 22$, compared to 50 GDMD-modes. An optimal configuration of γ requires some knowledge about the particular sensor and its noise component. In particular, error prone sensors benefit from smaller γ values while reliable measurements can utilize higher values. As a rule of thumb, $\gamma \approx 1.0$ has proven to produce relatively good results for most of the sensors utilized in the context of this thesis. A more detailed depiction of various DMD implementations can be found in previous publications [6][4].

In order to investigate the applicability of DMD to robotic behavior the GDMD and SDMD are compared to several classical interpolation schemes. For this, the NAO robot's longitudinal walking behavior is executed with five equidistant step lengths $\langle -4, -2, 0, 2, 4 \rangle$ [cm] and recorded for four seconds. The resulting 400 samples of proprioceptive measurements are separately stored in $\mathbf{Z}$. For example, $\mathbf{Z}_{\mathbf{c_x^m}}$ has five columns where each column contains 400 measurements of the c_x^m and is visualized in Figure 5.9 left. Given these samples, the goal was to generate c_x^m measurements for step lengths that were not recorded during training with an accuracy of 0.01 [cm].

To this end, the GDMD and SDMD are compared with classical methods, e.g., near-

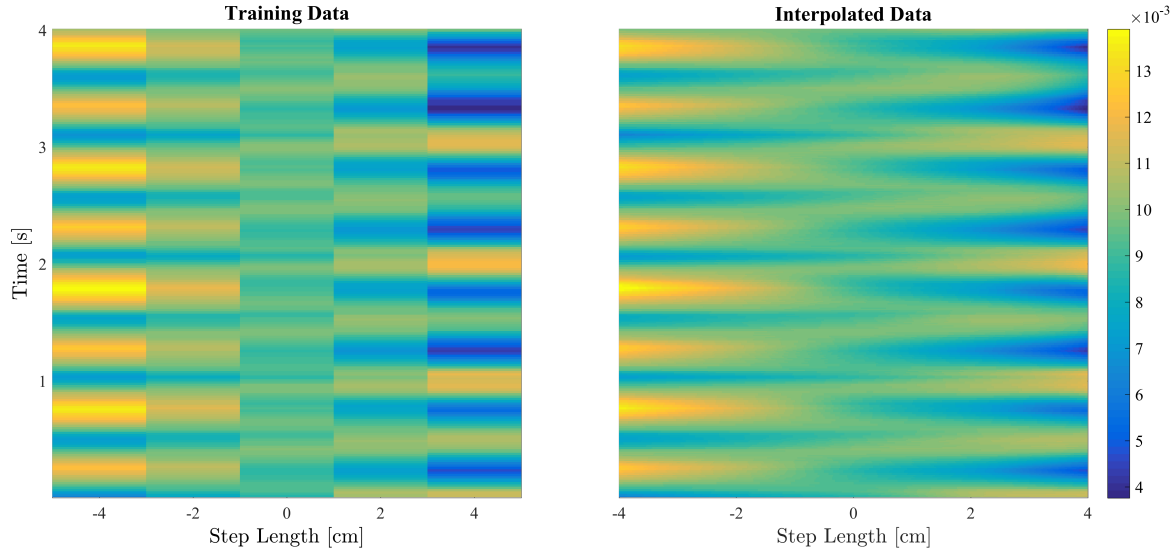


Fig. 5.9: GDMD is used to generate a model of the NAO robot's longitudinal center of mass c_x^m for its walking behavior. Hence, new c_x^m patterns are generated for unknown parameter settings. Left: The training data which consists of five equidistant c_x^m patterns during walking. Right: The c_x^m is interpolated with an interval of 0.01 [cm] resulting in approximated patterns for 800 possible step length configurations.

est neighbor, spline and bicubic interpolation. The accuracy of the interpolated data is evaluated by recording an additionally set of validation examples. These examples contain proprioceptive measurements for the so far unknown walking step lengths $\langle -3, -1, 1, 3 \rangle$ [cm].

The first row of Table 5.1 summarizes the resulting Mean Relative Errors (MREs) for the c_x^m. As can be seen, the GDMD, which utilizes the maximum of four modes, scores the lowest MRE among all methods. In contrast, SDMD considers the influence of noise and reduces the number of modes to three. This results in a less accurate model but still outperforms the classical interpolation schemes. Figure 5.9 right shows the interpolation result achieved by utilizing GDMD.

As already mentioned, the training data originates from a robot's standard proprioceptors which may be affected by strong noise. Hence, forcing a low number of non-zero amplitudes in $\boldsymbol{\alpha}$ can reduce the influence of disturbances in the approximation. To give an example of a noisy low-cost proprioceptor, the robot's longitudinal acceleration[1] a_x is interpolated. The MRE of the interpolation results can be seen in the second row of Table 5.1. Again, GDMD uses all of the extracted modes. Since some of these modes mainly contain noise this results in a corrupted interpolation what deteriorates the performance to a similar MRE as for the classical interpolation schemes. In contrast, SDMD concentrates on the modes that best approximates the underlying dynamics of the intrinsic proprioception. In this case, one mode was set to zero which apparently contained strong noise. Hence, SDMD exhibits a better interpolation quality than the GDMD approach for rather noisy sensors.

[1]The acceleration is measured by an inertial measurement unit integrated in the NAO robot's chest.

Tab. 5.1: The MRE of the validation examples with regard to the corresponding interpolation results of GDMD, SDMD and several classical interpolation schemes. Here, GDMD shows the highest accuracy for the c_x^m while SDMD performs best for the noisy a_x.

	nearest neighbor	**spline**	**bicubic**	**GDMD**	**SDMD**
c_x^m	4.42	4.62	4.25	**2.43**	3.36
a_x	11.23	16.95	15.3	16.78	**7.4**

5.2.3 Scaled Feature Space

First, a low-dimensional feature space is computed from the non-interpolated training data. Here, PCA is applied to the recorded proprioceptive measurements $\mathbf{X}$. In order to dampen noisy proprioceptors, the PCs with the lowest eigenvalues are erased. The acquired training data $\mathbf{X}$ is then projected into this feature space which results in a set of low-dimensional trajectories

$$\mathbf{X}' = \begin{bmatrix} {X'}_{1,1} & \cdots & {X'}_{1,m'} \\ \vdots & \ddots & \vdots \\ {X'}_{n\cdot k,1} & \cdots & {X'}_{n\cdot k,m'} \end{bmatrix}, \tag{5.12}$$

where $n \cdot k$ is the overall amount of training samples and $m' \leq m$ is defined by the remaining PCs. Hence, each column $\mathbf{x}'[1], \ldots, \mathbf{x}'[m']$ represents the temporal evolution of one low-dimensional trajectory. A drawback of this procedure is that possibly beneficial dependencies between proprioceptors and behavior parameters, e.g., correlations between joint angles and the walking step length, are neglected. In the following, correlations between the proprioceptors and the behavior parameter are uncovered by means of information theory.

For this, PCs with a strong statistical coherence to the behavior parameter configuration are detected. To this end, the information transfer between the behavior parameter $\mathbf{y}$ and the projected training data trajectories $\mathbf{X}'$ is taken into account. A PC is deemed relevant, if changes in the behavior parameter configuration are a good predictor for the activity of the corresponding trajectory. From an information-theoretic point of view, this type of relationship can be detected by using Transfer Entropy (TE) (see Section 3.1.3 p. 32). More precisely, $TE_{\mathbf{y}\rightarrow\mathbf{X}'} = I(\mathbf{X}' + 1; \mathbf{y}|\mathbf{X}')$ quantifies the incorrectness of the assumption, that in the absence of information transfer from $\mathbf{y}$ to $\mathbf{X}'$, the state of $\mathbf{y}$ has no influence on the transition probabilities on $\mathbf{X}'$.

Computing the TE requires the calculation of the conditional as well as joint probability of co-occurrences in $\mathbf{y}$ and $\mathbf{X}'$. To estimate these probabilities, without resorting to density estimation, the projected trajectories of $\mathbf{X}'$ are preprocessed by two subsequent steps: normalization and discretization (see Section 2.2). Figure 5.10 shows a simple example for the preprocessing of the first PC trajectory (blue) with regard to the step length parameter (green) of walking. For this, a total of twenty-one equidistant step lengths was recorded. First, the original range of each PC trajectory is

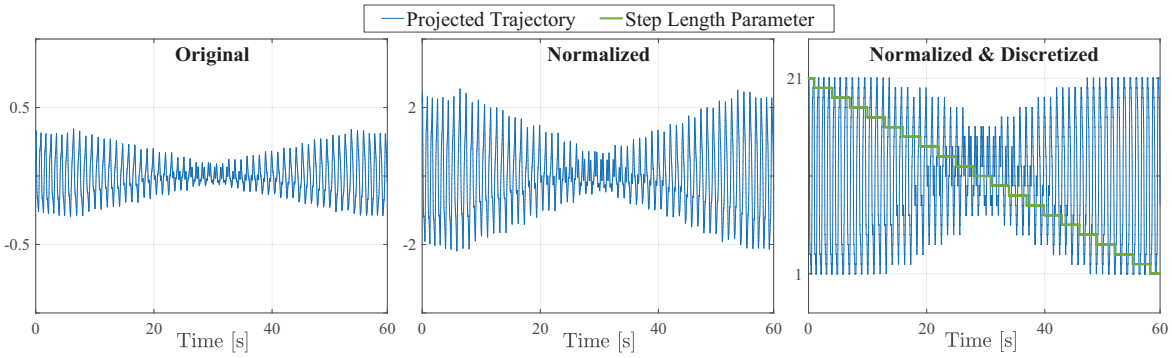


Fig. 5.10: The projected training data trajectory of the first PC is normalized and discretized to increase the efficiency of information-theoretic measures. Left: The original PC of a 60 second longitudinal walking behavior are configured with twenty-one equidistant step lengths. Middle: The training data trajectory is normalized by utilizing the z-score. Right: The continuous space of the normalized values is further discretized to a fixed number of states. Here, the number of states is equivalent to the states of the particular behavior parameter, e.g., the twenty-one step length configurations.

normalized by utilizing the standard score which is also known as z-score. Next, the continuous space of the normalized dimension is discretized. Here, the number of quantization levels is equivalent to the corresponding behavior parameter space to result in twenty-one states. This has the advantage of stability and independence of input value transformations.

Variations of the robot's behavior parameter configuration could possibly have a time delayed impact on proprioceptors and consequently on the projected trajectories. The so far utilized TE algorithm only allows to draw conclusions based on transitions of a one-step delay between $\mathbf{y}$ and $\mathbf{X}'$. A more general approach is implemented by using delayed TE $TE_{\mathbf{y}\to\mathbf{X}'}(\boldsymbol{\delta})$ (see Equation 3.16 p. 35) which introduces multiple possible delays $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_j)$. Here, the peak values between the particular behavior parameter $\mathbf{y}$ and the projected trajectories $\mathbf{x}'[1], \ldots, \mathbf{x}'[m']$ are calculated by

$$TE^*_{\mathbf{y}\to\mathbf{x}'[i]}(\boldsymbol{\delta}) = \text{argmax}_{\forall j}\, TE_{\mathbf{y}\to\mathbf{x}'[i]}(\delta_j). \tag{5.13}$$

Each original PC is then scaled by the calculated peak values such that components with higher TE are stretched and those with lower TE are shrunk. The resulting scaled feature space, is utilized to project original and interpolated training data as well as actual measurements and is in the core of this lazy I-BSPM implementation.

5.2.4 Behavior Parameter Estimation

During task execution the measured proprioception $\hat{\mathbf{x}}$ is projected into the scaled feature space. The resulting low-dimensional point is defined as $\hat{\mathbf{x}}'$. In contrast to the previous implementation, the estimated behavior parameter cannot be derived from these projections directly. This is due to the fact, that DMD techniques interpolate unknown proprioceptive measurements from the training data but provide no mapping technique. Instead, a separate approach which finds the optimal mapping between the actual proprioception and the training data is introduced. More precisely, a $\hat{n}$-length

sequence of the actual proprioceptive measurements $\hat{\mathbf{X}}$ is projected into the scaled feature space upon online behavior execution. The resulting low-dimensional trajectory $\hat{\mathbf{X}}'$ is compared with the trained and interpolated trajectories in order to identify the best matching and therefore most similar behavior parameter. The basic idea is that the I-BSPM maps the behavior parameter, which was configured for the most similar training data, as an estimate of the perceived proprioception.

For this, Dynamic Time Warp (DTW) [106] a time series alignment algorithm for measuring the similarity between two temporal sequences, is utilized. In particular, the goal is to find the optimal correspondence between projected training $\mathbf{X}'$ and runtime data $\hat{\mathbf{X}}'$ which have the following structure

$$\mathbf{X}' = \begin{bmatrix} \mathbf{x}'(1) \\ \vdots \\ \mathbf{x}'(n) \end{bmatrix}, \hat{\mathbf{X}}' = \begin{bmatrix} \hat{\mathbf{x}}'(1) \\ \vdots \\ \hat{\mathbf{x}}'(\hat{n}) \end{bmatrix}. \tag{5.14}$$

Due to the significant difference in length $\hat{n} \ll n$ the task is formulated as finding a subsequence

$$\mathbf{X}'(s^* : e^*) = (\mathbf{x}'(s^*), \mathbf{x}'(s^* + 1), \ldots, \mathbf{x}'(e^*)) \tag{5.15}$$

with $1 \leq s^* \leq e^* \leq n$, where s^* is the starting index and e^* is the end index that optimally fits to the corresponding subsequence $\hat{\mathbf{X}}'$. This technique is also known as Subsequence Dynamic Time Warping (SDTW) [122]. To find the optimal subsequence the accumulated cost matrix $\mathbf{D}$ is determined as

$$\begin{aligned} \mathbf{D}_{i,1} &= \sum_{g=1}^{i} c(\hat{\mathbf{x}}'(g), \mathbf{x}'(1)), i \in [1 : \hat{n}] \\ \mathbf{D}_{1,j} &= c(\hat{\mathbf{x}}'(1), \mathbf{x}'(j)), j \in [2 : n] \\ \mathbf{D}_{i,j} &= min\{\mathbf{D}_{i-1,j-1}, \mathbf{D}_{i-1,j}, \mathbf{D}_{i,j-1}\} + c(\hat{\mathbf{x}}'(i), \mathbf{x}'(j)), \end{aligned} \tag{5.16}$$

where c is a local distance measure which usually is defined as the Euclidean Distance (EUD). As a result the SDTW algorithm is able to determine the path with minimal costs ending at (e^*, n), where e^* is given by

$$e^* = \underset{e \in [1:n]}{\operatorname{argmin}} \mathbf{D}_{\hat{n},e}. \tag{5.17}$$

To determine the warping path $\mathbf{p}^* = (p_1, \ldots, p_L)$, which starts at $p_1 = (1, s^*)$ and ends at $p_L = (\hat{n}, e^*)$, a dynamic programming recursion is used. The resulting path $\mathbf{p}^*$ represents the optimal subsequence of $\hat{\mathbf{X}}'$ in $\mathbf{X}'$.

The behavior parameter which was configured while producing these similar proprioceptive measurements during training y_{p_L} defines the output of the I-BSPM. An extrinsic perturbation is then inferred from the difference between this estimate and the actually configured behavior parameter. This has the advantage that a perturbation is measured in the behavior parameter space what simplifies an adequate behavior adaptation. In particular, the estimated behavior parameter represents an appropriate

behavior adaptation and is utilized as setpoint. Hence, no predefined reaction rules are required anymore.

In the following, the interaction scenario with the humanoid NAO robot, as already examined in the previous experiments (Section 5.1.5), is investigated. This shows that an I-BSPM achieves similar and even superior results while requiring less user effort and minor knowledge about the particular robot platform. A video presenting some of the experiments conducted in this section can be found under the following link: `https://youtu.be/wHZYx6Dzswk`.

Afterwards, the usability of the human-robot interaction task is analyzed by conducting a user study. For this, a questionnaire which is suitable for small sets of test users is utilized. Two groups which divide users depending on their experience with humanoid robots are analyzed. By comparing the results, this allows to analyze the intuitive applicability of the proposed I-BSPM implementation.

5.2.5 Evaluation

In this section several experiments are conducted without the use of higher level stability parameters, e.g., center of mass. This increases the applicability of the presented I-BSPM approach to non-experts. Furthermore, the lazy I-BSPM implementation does not contain an abstract proprioception model and instead requires a large number of training examples. Here, arbitrary accurate training examples are generated from few examples by utilizing the examined GDMD method for interpolation purposes.

The objective of the presented application is to estimate the strength and direction of extrinsic perturbations caused by a human interaction partner during longitudinal walking. The I-BSPM utilizes 54 seconds of raw sensor readings rather than stability parameters. More precisely, recorded proprioceptors include 24 joint angle values and raw foot pressure measurements for nine equidistant step length configurations: $\langle 4, 3, 2, 1, 0, -1, -2, -3, -4 \rangle$ [cm]. By utilizing the proposed GDMD interpolation technique, the overall accuracy of the training examples is increased to 0.01 [cm]. In contrast to SDMD, no additional user effort for an adequate configuration is required what ensures independence from expert knowledge about the particular sensor readings. Furthermore, SDMD is less beneficial since the subsequent PCA procedure already dampens noisy inputs by erasing less variant components. In particular, a four-dimensional feature space of the angle sensors and a six-dimensional feature space of the pressure sensors is calculated retaining 95 % of the variance contained in the non-interpolated training examples.

These feature spaces are scaled by their information transfer received from the step length parameter. This is necessary since PCA extracts the most characteristic properties of a behavior but does not utilize dependencies with the behavior parameter configuration. TE therefore measures how strong each PC depends on the configured step length parameter. Here, only the trajectories of the non-interpolated PCs are normalized and discretized to nine states and utilized for computing the TE. In order to include delayed dependencies, the peak TE values for a time window which con-

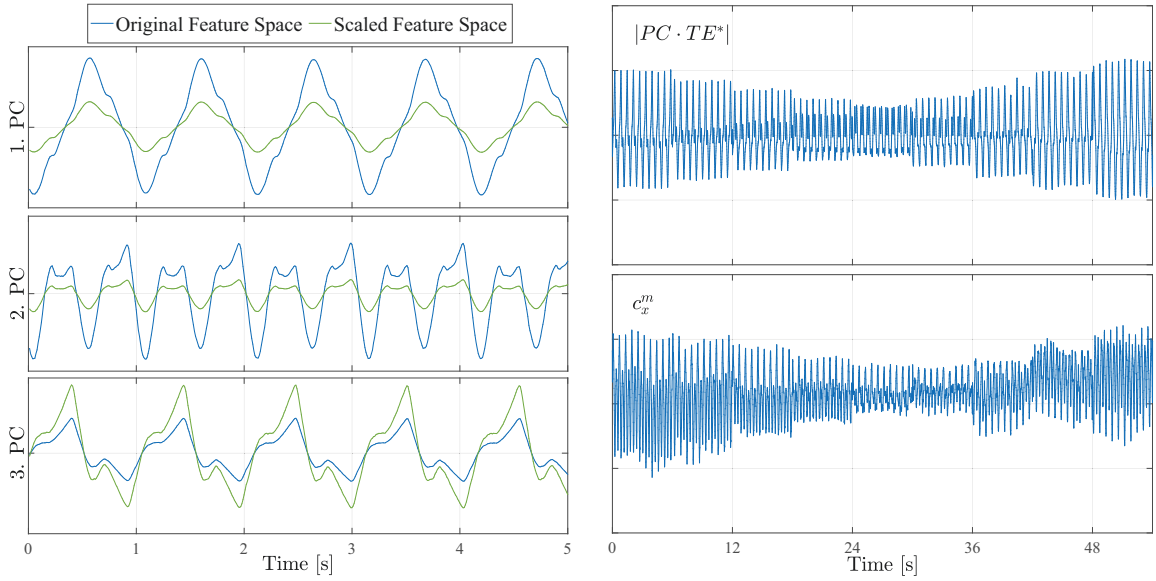


Fig. 5.11: The original PC trajectories of longitudinal walking are scaled by their information transfer received from the step length parameter. Left: The third PC has the highest TE value and is stretched while the first and second PC are dampened. The scaled PCs comprise the feature space of the I-BSPM$_{\text{angle}}$. Right: The vector length of $PC \cdot TE^*$ compared to the measured stability parameter c_x^m. In contrast to utilizing such higher level stability parameters, the proposed I-BSPM does not need any knowledge about the robot's kinematics or mass distribution to generate a comparable result.

siders the last ten measurements are calculated (see Equation 5.13 p. 80). PCs that have a high TE w.r.t. the robot's behavior are deemed more influential and relevant. Figure 5.11 left shows some of the original (blue) and scaled (green) PCs which are calculated from the recorded joint angle values. While the first, second component are dampened, the third one is stretched. The resulting scaled feature spaces represent the corresponding I-BSPM and are referred to as I-BSPM$_{\text{angle}}$ and I-BSPM$_{\text{pressure}}$. During runtime behavior execution, these I-BSPMs are applied to estimate the presence and amount of extrinsic perturbations.

For this, an extrinsic perturbation is detected by comparing the low-dimensional training examples with the projection of the measured proprioception. SDTW is used as a distance function in order to include the temporal pattern for this comparison. For the I-BSPM$_{\text{angle}}$, the influence of the third PC on the SDTW algorithm is increased while decreased for the other components. Furthermore, Figure 5.11 right shows the vector length of each PC scaled with its particular TE $|PC \cdot TE^*|$ compared to the longitudinal center of mass c_x^m. As can be seen, the properties of I-BSPM$_{\text{angle}}$ are remarkably similar to the c_x^m, without actually having to provide prior knowledge about the robot kinematics or mass distributions.

Both I-BSPMs are utilized for the detection of human guidance during runtime execution of the walking behavior. In order to ensure better comparability, perturbations are applied to different parts of the robot. Figure 5.12 shows the parameter estimations of I-BSPM$_{\text{angle}}$ and I-BSPM$_{\text{pressure}}$ for a constant parameter configuration and various perturbations. Perturbation (a) is recognized by both I-BSPMs, because the angles as well as the pressure sensors are affected. If no extrinsic perturbation is recognized dur-

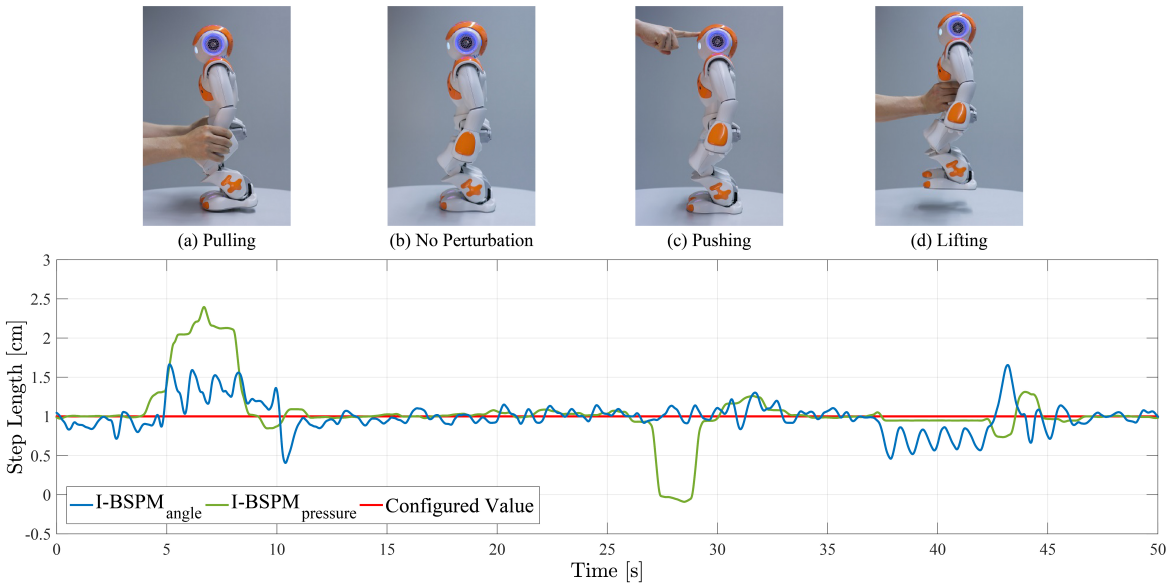


Fig. 5.12: Strength and direction of extrinsic perturbations are measured by the difference between estimated and configured behavior parameter. Certain I-BSPMs help to recognize and qualify a variety of perturbations. (a) is detected by the I-BSPM$_{\text{angle}}$ and the I-BSPM$_{\text{pressure}}$ while perturbations in (c) and (d) are detected by only one of these models.

ing (b), the parameter estimations of both I-BSPMs give a close approximation of the configured parameter value. However, in (c) the perturbation affects the pressure sensors rather than angles and in consequence can only be measured by the I-BSPM$_{\text{pressure}}$. Finally, perturbation (d) leads to a measurement of flat zeros for the pressure sensors. This is not part of the training data and, consequently, cannot be recognized by the I-BSPM$_{\text{pressure}}$. Hence, multiple I-BSPMs which are trained for different proprioceptive measurements are suitable to qualify certain perturbations, allowing conclusions about perturbation characteristics.

It can be argued that the interpolation and mapping technique is applicable without the need of PCA and TE. Hence, mapping a sequence of proprioceptive runtime measurements can be done directly in the original (high-dimensional) proprioception space. Indeed, this is possible but as will be shown results in a significant loss of accuracy. Here, a validation data set was recorded while the robot frequently reduced its step length. More precisely, the robot walked for one minute and step lengths within between $(4, \ldots, -4)$[cm] resulting in 6000 validation samples. Table 5.2 summarizes the Mean Absolute Error (MAE) of these samples when utilizing I-BSPM$_{\text{angle}}$ or I-BSPM$_{\text{pressure}}$.

As can be seen, both I-BSPMs are not applicable in the high-dimensional space of the original measured proprioceptors. Here, I-BSPM$_{\text{angle}}$ achieves clearly better results when utilizing exclusively PCA. This is due to the fact that the joint angles are especially influenced by spurious relationships, which are strongly dampened by PCA, even without TE. By combing PCA and TE the lowest MAE is achieved. Furthermore, as shown by I-BSPM$_{\text{pressure}}$, using only PCA can obfuscate behavior parameter correlations. After appropriate scaling of the corresponding PCs the I-BSPM$_{\text{pressure}}$ mapping

Tab. 5.2: The MAE of the I-BSPM estimation results (in [cm]) when utilizing joint angles or pressure values for all permutations of PCA and TE. The $\text{I-BSPM}_{\text{angle}}$ accuracy strongly benefits from PCA while the usage of TE is vital for the $\text{I-BSPM}_{\text{pressure}}$. As can be seen, combining PCA and TE results in the lowest MAEs.

	¬PCA¬TE	**¬PCA TE**	**PCA¬TE**	**PCA TE**
$\text{I-BSPM}_{\text{angle}}$	0.29	0.28	0.06	**0.02**
$\text{I-BSPM}_{\text{pressure}}$	0.25	0.32	0.3	**0.04**

accuracy increases. To conclude, a combination of PCA and TE results in the lowest MAE for both I-BSPMs and therefore is vital for the applicability of the presented implementation.

5.2.6 Usability Analysis

A user study is conducted in order to investigate the usability of the presented I-BSPM implementation. For this, the well-known System Usability Scale (SUS) created by Brooke [123] is used. SUS is a ten-item questionnaire for quickly measuring a system's usability with a score in range from 0 to 100. As noted by Bangor, Kortum, and Miller [124], the SUS questionnaire further is a method to effectively determine a system's usability even with a smaller number of test subjects.

In particular, 20 male and 9 female subjects (ages $13-18$, mean $= 15.28$) participated in this user study. All participants were high-school students and were divided into two groups. The first group consisted of 21 students who had never before interacted with the NAO robot. This group is referred to as *beginners*. The second group containing 8 students had participated on an one day NAO workshop before and, thus, is referred to as *experts*. The total time per participant including instructions, experiment and questionnaire was about ten minutes.

In this experiment, the $\text{I-BSPM}_{\text{angle}}$, which derives proprioception only from angle values, was evaluated. Subjects had to steer the robot by physically touching and guiding it to several target positions. As starting condition, the robot is walking in place with a step length of zero centimeters. Next, the subjects were asked to steer the robot along its longitudinal axis to three specified points on the floor by physically pushing the robot forwards and backwards, as illustrated in Figure 5.13 left. In order to accomplish this task, a forward-to-backward and a backward-to-forward transition had to be completed. While walking towards the target points, the robot could be accelerated and decelerated by the difference between the estimated and actually configured step length parameter.

While walking backwards, the center of gravity is strongly moved towards the front of the robot. Moreover, this effect is further increased by the human guidance. In consequence, a backward-to-forward transition stronger influences the robot's stability and is more challenging than a forward-to-backward transition. Considering this, the first waypoint was defined in the front of the robot, the second behind it and the third

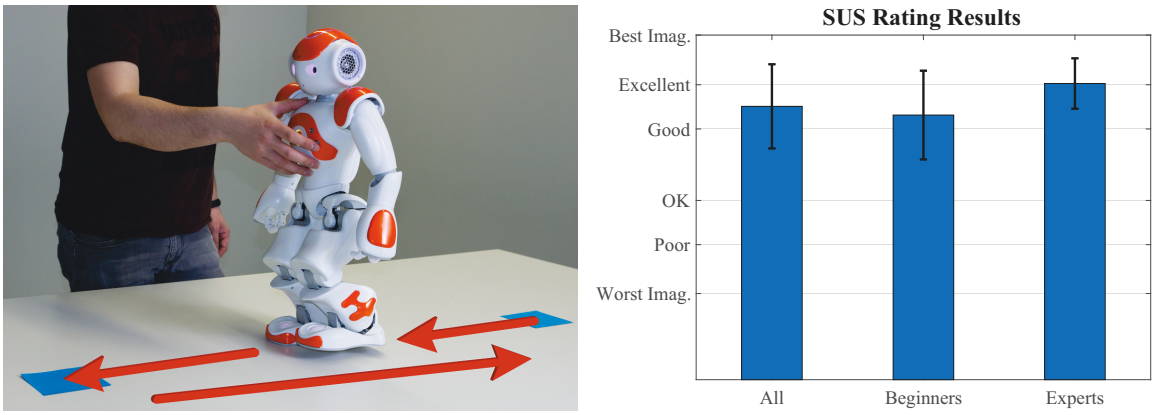


Fig. 5.13: The usability of the presented I-BSPM implementation is investigated by conducting a user study. Left: The experimental conditions. First the subjects have to lead the robot forward to the blue marker and fulfill a forward-to-backward transition. Next, they have to lead the robot backward to another blue marker and fulfill a backward-to-forward transition. Finally, they have to lead the robot back to its initial position. Right: The results of the SUS questionnaire. The usability depends on the user's knowledge about the robot platform. The overall rating is between good and excellent what confirms the intuitivity and usability of the proposed approach.

was the robot's start point. This design results in an increasing interaction difficulty during the experiment.

As suspected most of the participants perceived the backward-to-forward transition as more challenging what often led to falling over of the robot. In this case, the subject was allowed to try the complete experiment once again. During the second execution nearly all subjects were able to successfully execute both walking transitions. This fast learning curve is attributed to a high intuitivity of the proposed approach.

Statistical analysis of the SUS questionnaire further underlines this conclusion. For this, the mean and standard deviation of the SUS score for the beginners, the experts and all participants is computed and compared in respect to the rating proposed by Bangor, Kortum, and Miller [124]. Hence, the SUS score of 0 to 100 is assigned to values between *worst* and *best imaginable*. As shown in Figure 5.13 right the experts assign an *excellent* mean score while the beginners evaluate the approach with a *good* mean score. The better rating by the experts can be explained by their familiarity with the robot hardware and that they have less fear of contact than the beginners. This observation is confirmed by the measured standard deviation which is smaller for the experts than for the beginners. In general, this user study confirms a *good* to *excellent* usability and a steep learning curve.

5.2.7 Conclusion

In this section, an I-BSPM was applied to accurately identify human physical influences on a robot. An overview of the proposed lazy learning is given in Figure 5.14. Here, no abstract model is utilized to map arbitrary inputs to their most probable outputs. Instead, the similarity between training examples and runtime data is used to estimate the actual behavior parameter.

More precisely, regular training examples are acquired for a few equidistant param-

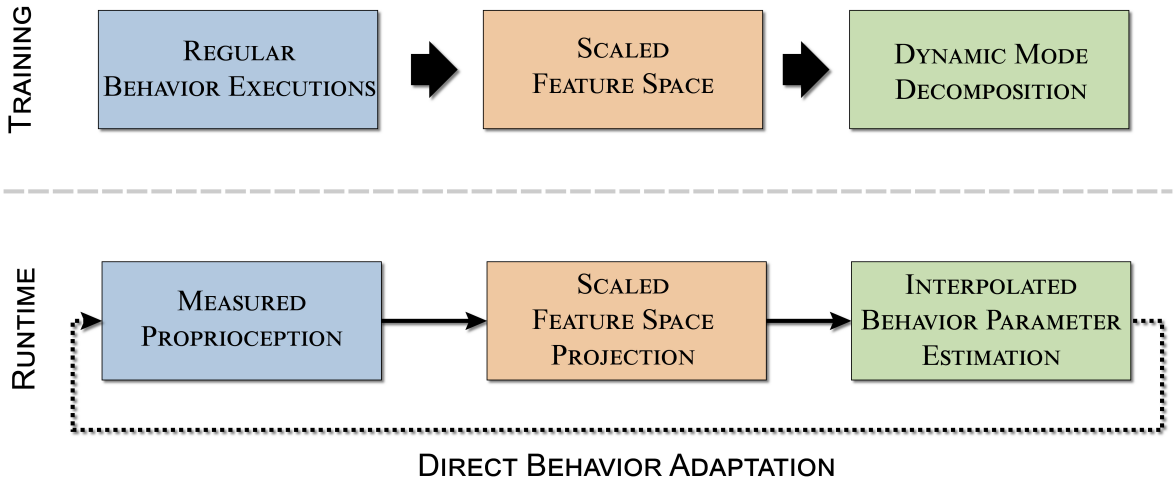


Fig. 5.14: Overview of the presented lazy I-BSPM approach with regard to the methodology introduced in Section 4.3. Training data, acquired from proprioceptors during regular execution of the behavior, is processed to a scaled feature space by utilizing PCA and TE subsequently. Here, DMD is utilized to interpolate unknown proprioceptive measurements for arbitrary behavior parameter configurations. During runtime, a sequence of the measured proprioception is being projected into the scaled feature space and mapped to the nearest training samples by utilizing the SDTW algorithm. The behavior parameter configuration of the most similar training example is then used as an estimate of the behavior parameter. Utilizing this estimate as setpoint results in a direct behavior adaptation.

eter configurations. The recorded proprioceptors are then projected into a feature space by utilizing the most relevant components (PCA). Here, each component is further scaled by the information transfer (TE) received from the particular behavior parameter. Hence, the influence of more relevant components on the utilized mapping technique is strengthened. At runtime, the measured proprioception is projected into the scaled feature space and compared to the training samples by utilizing the SDTW algorithm. In particular, the training example with the smallest EUD to the measured proprioception is detected. The parameter configuration of the closest match is then used as an estimate of the actual behavior parameter.

As stated above, only a few training examples for equidistant parameter configurations are recorded what, without further interpolation, would result in a limited output accuracy. In order to generalize outputs for arbitrary inputs the DMD, a novel method from the field of fluid dynamics, was applied. DMD isolates the dynamics of a nonlinear system and in consequence is well suited to separate noise from intrinsics. For this, periodic robot behaviors, e.g., walking, are considered to have similar properties as flows. One can argue that not all behaviors have a periodic nature. In most of the cases, this can be solved by periodically performing an inverse behavior execution after the original one, e.g., knee bending. As a result, the recorded proprioceptive measurements can be interpolated arbitrarily between equidistant training examples. During runtime, a sequence of the measured proprioception is compared to the recorded and interpolated training data. This enables the proposed approach to function with only few training examples while providing sufficient estimation accuracy.

5.3 Summary

In order to infer guidance information during cooperative human-robot interaction, two BSPM applications for measuring extrinsics during behavior execution were presented. For this, the expected proprioception, which is estimated from knowledge about regular behavior executions, is compared to the measured proprioception. The amount and direction of extrinsic perturbations can then be used to adapt the robotic behavior accordingly. More precisely, the goal was to adapt a humanoid robot's behavior to the forces induced by the human interactant such that the robot is following the human guidance. For this,

1. an eager approach which applies F-BSPMs and
2. a lazy approach which utilizes I-BSPMs were presented.

Here, the abstract model of (1) generalizes outputs by modeling a probability distribution over functions (GPR) while (2) introduces a non-linear interpolation scheme over periodic training data (DMD). As a result, both implementations required only few training examples with a maximum recording length of one minute. This is vital to ensure the practical applicability of the BSPM approach.

The acquired training data is then used to learn the particular BSPM. The F-BSPM predicts the proprioception of the most probable stability parameter where extrinsics are defined by the difference between measured and expected stability. The I-BSPM utilizes the configuration of the most similar training example in interpolated space as an estimate of the actual behavior parameter. Here, extrinsic influences on the proprioception are derived from the difference between the actually configured and estimated behavior parameter.

Hence, (1) measures the strength and direction of perturbations inside the stability parameter space. This requires the definition of precise rules which map a perturbation to an adequate behavior adaptation. In contrast, (2) measures perturbations inside the parameter space. This allows utilizing the estimated behavior parameter as setpoint for a direct behavior adaptation.

A further advantage of (2) is the automatic identification of relevant training data by utilizing the information transfer between the behavior parameter and the proprioceptors. Here, available proprioceptive information is leveraged resulting in patterns which are similar to the utilized stability parameters. Hence, information-theoretic measures are proved beneficial for the application in BSPMs and are further investigated in the following chapters. Furthermore, the generalizability of the proposed lazy I-BSPM approach will be demonstrated by investigating a tool usage application for an industrial robot platform.

6 Augmenting Robotic Proprioception with Virtual Force Sensors

This chapter investigates the applicability of the proposed BSPM approach on an industrial robot where the presented applications are focusing on the physical interaction with the environment. The idea is to give the robot similar capabilities as skilled human workers who are able to estimate exact forces from prior experience only. Concretely, the proprioception of a UR5 robotic arm is augmented with accurate force sensing capabilities without the need of special purpose sensors.

In practice, the firmware of such industrial robots often provides some kind of force-torque estimation interface where a major drawback is its limited accuracy. Here, the estimation accuracy depends on a precise configuration of the underlying mathematical model which needs to be provided by the user. For example, the UR5 robot includes a built-in firmware which estimates force-torque values acting at its Tool Center Point (TCP). The manufacturer specifies a force accuracy of $\pm 25\,\mathrm{N}$ if dimensions, orientation, and weight of the tool are configured correctly. Furthermore, the manufacturer warns that the utilized force estimation algorithm provides no protection against momentum. While not relying on a manual configuration, one goal of the proposed V-BSPMs is to achieve higher estimation accuracy than this built-in firmware.

Another possible solution is to expand the robot with a dedicated sensor that measures Force-Torque (FT) values at its TCP. These FT sensors are appropriate to solve a wide variety of tasks but also have a negative impact on costs. Furthermore, the additional weight limits the robot's payload. With regard to rough environmental conditions, such FT sensors usually provide no protection against dust and spray water where an additional external cover intensifies these issues. Hence, utilizing built-in firmware or FT sensors results in major restrictions which reduce the general and intuitive applicability of the particular robot platform. To solve this without the need of additional hardware or expert knowledge, the BSPM approach can be applied in the virtual sensor mode. The corresponding V-BSPM augments the robot's proprioception but does not detect extrinsic perturbations (see Section 4.3). In particular, two different implementations of the V-BSPM are presented and applied to specific tasks:

1. A lazy V-BSPM estimates the tightening torque of a custom wrench by detecting the most similar training example.

2. An eager V-BSPM approximates the weight of a water extraction station by memorizing long-term dependencies between proprioceptive measurements.

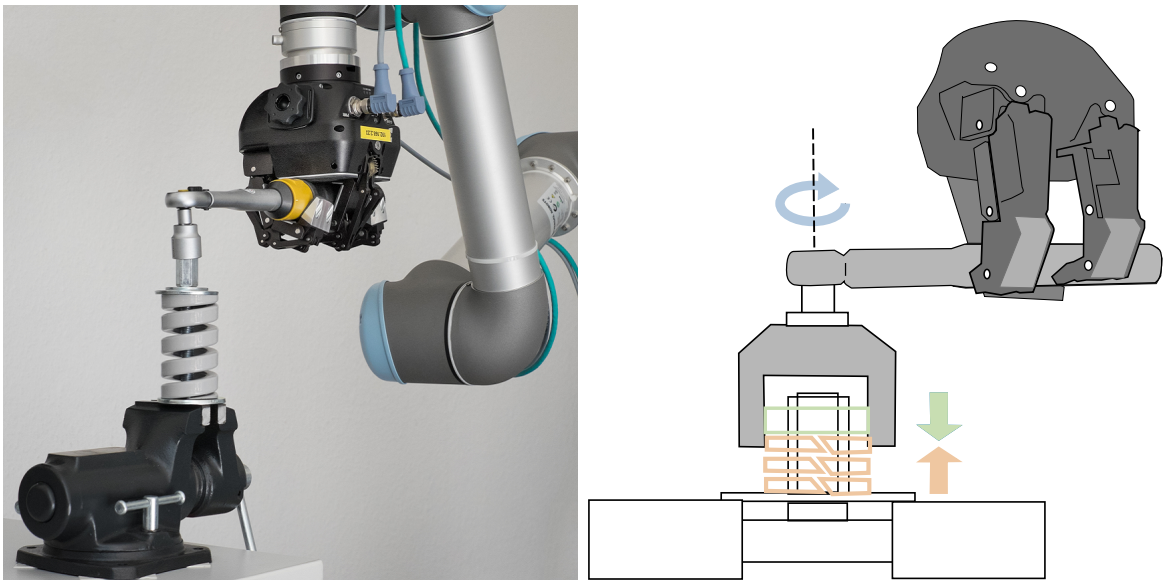


Fig. 6.1: A V-BSPM is applied to augment the proprioception of a UR5 robotic arm with tool usage capabilities. Left: A three-fingered gripper mounted on a UR5 robotic arm is utilizing a usual torque wrench. The torque exerted to a screw nut is estimated from the robot's standard proprioceptors. Right: The robot turns the torque wrench about 45°. The exerted torque (blue) results in a preload force (green) which is countered by the force conducted by a pressure spring (orange).

For this, training examples of the UR5 robot's standard proprioceptors are further augmented by context descriptions or labels. At runtime, the robot utilizes the V-BSPM to classify the actual context label, e.g., the torque or weight exerted to the TCP. As will be shown, these V-BSPMs achieve an accuracy which is comparable to and even beyond state-of-the-art FT sensors.

6.1 Lazy V-BSPM Learning

In this section, the V-BSPM approach is applied to the UR5 robotic arm which conducts the physical interaction of tightening a screw nut (see Figure 6.1 left). Here, a three-fingered gripper is mounted on the UR5 robotic arm to limit a given tightening torque when configuring the screw nut with a custom wrench. This is motivated by a trained mechanics ability to estimate a screw nut's tightening torque from prior experience only. The elaborated setup is further explained in Figure 6.1 right. Here, the exerted torque (blue) results in a preload force (green) which is countered by the force conducted by a pressure spring (orange).

The first step of the V-BSPM is to acquire a small set of training examples which represent the intrinsics of the behavior. For this, a 45° tightening movement is recorded multiple times for equidistant torque configurations. Here, the configured torque is used as label of the corresponding training example. This ensures that correlations between proprioceptors and the torque are contained in the training data. The V-BSPM is an expanded version of the lazy algorithm which was introduced in Section 5.2. Hence, the number of training examples can be increased by utilizing the introduced DMD interpolation scheme from the field of fluid dynamics (see Section 5.2.2 p. 75).

The recorded behavior examples are used to generate two particularly different feature spaces. One focuses on temporal correlations while the other utilizes correlations to the labels of the torque exerted to the screw nut. For the proposed task, the first feature space is used to identify the actual phase of the behavior execution. Here, the phase determines the actual state of the behavior which is strongly related to the relative execution time. One could even argue that the relative execution time can be used to determine the robot's state. This may be possible when a behavior is always guaranteed to be executed in exactly the same way. In the case of the UR5 robot, each behavior execution has a varying onset and consequently the relative execution time does not precisely describe the state of the behavior. Hence, a precise phase estimation procedure is vital. The second feature space then determines the exerted torque which benefits from a search space reduction of the selected phase. To this end, runtime measurements are projected into the corresponding feature spaces and compared to the training trajectories. Finally, the label which is assigned to the training example with the highest similarity is used as an estimate of the exerted torque. In the following, each step of the lazy V-BSPM implementation is explained in more detail.

6.1.1 Data Acquisition: Behavior Examples with Context Labels

The real-time interface of the UR5 robot provides access to 105 sensor readings. As commonly provided by most robot platforms, these can be read in a constant interval. In particular, the UR5 robot provides equidistant measurements every 8 ms, i.e., at a frequency of 125 Hz. The proprioceptors provide a wide range of measurements which also contain less useful information such as the main board voltage and redundant data as control, target and actual joint values.

As stated above, the interface also provides estimates of the force-torque values at its TCP where alleged benefits depend on a task-specific configuration. With a correct configuration the firmware's torque accuracy for the utilized wrench is $\pm 25\,\mathrm{N} \cdot 0.23\,\mathrm{m} = \pm 5.75\,\mathrm{N\,m}$. To reduce the user effort and to ensure independence from expert knowledge this configuration is not provided in the context. This further confuses the firmware and results in unreliable torque estimates. Hence, all sensor readings are recorded for a subsequent selection of the most significant proprioceptors.

For this, the $m = 105$ proprioceptors are recorded during performing the 45° tightening movement. The behavior execution took two seconds resulting in $n = 250$ equidistant samples $\mathbf{x} = (x_1, \ldots, x_m)$. Additionally, the relative time w and the configured torque v are used to label the context for each m-dimensional recording. Hence, a recording for one torque configuration has the following appearance

$$\mathbf{X}_1 = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}, \mathbf{v}_1 = \begin{pmatrix} v_1 = v \\ \vdots \\ v_n = v \end{pmatrix}, \mathbf{w}_1 = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix} \tag{6.1}$$

In contrast to the constant torque $\mathbf{v}_1$, the relative time $\mathbf{w}_1$ increases during behavior execution and is reset at the beginning of each behavior execution. Consequently, the

temporal pattern is uniform for different repetitions of the behavior.

Here, the tightening motion was recorded for ten different torque configurations. First, the screw nut was tightened with 6.0 N m and equidistantly increased by 1.0 N m up to 15.0 N m. Hence, the overall training data for $k = 10$ different torque configurations is defined by

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_k \end{bmatrix}, \mathbf{v} = \begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_k \end{pmatrix}, \mathbf{w} = \begin{pmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_k \end{pmatrix}. \tag{6.2}$$

Hence, the training data contains $n \cdot k = 2500$ samples of the robot's regular proprioception $\mathbf{X}$ together with the torque labels $\mathbf{v}$ and the relative execution time $\mathbf{w}$.

6.1.2 Proprioceptor Selection

The training data is investigated in order to detect phase and torque specific correlations with the proprioceptors. For this, two relevance values $\boldsymbol{\upsilon} = (\upsilon_1, \ldots, \upsilon_m)$ and $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_m)$ are computed for each proprioceptor.

- $\boldsymbol{\upsilon}$ describes how strong the proprioceptors are influenced by the torque $\mathbf{v}$.
- $\boldsymbol{\omega}$ describes how strong the proprioceptors are affected by the relative time $\mathbf{w}$.

In Section 5.2, Transfer Entropy (TE) has been used to solve related tasks for low-dimensional trajectories and achieved promising results. To this end, TE (see Section 3.1.3 p. 32) is used as a measure of predictability and information transfer between the relative time $\mathbf{w}$ or torque $\mathbf{v}$ to $j \in [1 \ldots m]$ proprioceptors $\mathbf{x}[j]$ by

$$\begin{aligned} \upsilon_j &= TE_{\mathbf{v} \to \mathbf{x}[j]} \\ \omega_j &= TE_{\mathbf{w} \to \mathbf{x}[j]}. \end{aligned} \tag{6.3}$$

For this, each proprioceptive measurement needs to be preprocessed by means of normalization (z-score) and discretization to a fixed number of states. Here, the calculation of $\boldsymbol{\upsilon}$ is based on the proprioceptive measurements $\mathbf{x}[j]$ which are discretized with ten states. In contrast, $\boldsymbol{\omega}$ makes use of a discretization with 250 states. This is due to the number of states in the corresponding recordings where $\mathbf{v}$ contains $k = 10$ labels of the torque and $\mathbf{w}$ the continuously increasing values for $n = 250$ time steps.

The proprioceptors which receive a high amount of TE depend stronger on the time or torque and are assumed to be beneficial for estimation purposes. Figure 6.2 left shows the resulting TE for the actual angle, velocity and current sensors contained in $\upsilon_{33}, \ldots, \upsilon_{50}$ and $\omega_{33}, \ldots, \omega_{50}$. For $\boldsymbol{\upsilon}$, the highest correlations are found in the group of proprioceptors which measure the actual joint currents while the joint angles and velocity sensors receive likewise less information. The joint current $\mathbf{x}[49]$ with peak TE value υ_{49} is visualized in the top of Figure 6.2 right. As can be seen, the proprioception differs for varying torques after a short time lag. This onset can be explained by the fact that the torque wrench is loosely connected to the gripper of the robot. Hence,

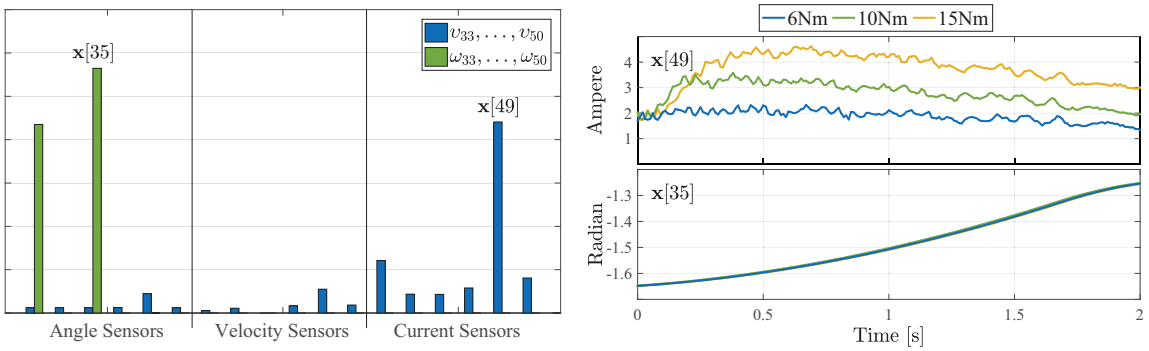


Fig. 6.2: Left: The normalized TE from the exerted torque $\boldsymbol{\upsilon}$ (blue) and the relative time $\boldsymbol{\omega}$ (green) to the robot's angle, velocity and current sensors. Right: The raw readings of the sensors with peak information transfer. Exerting different torques results in varying proprioceptive readings such as the current measurements of $\mathbf{x}[49]$ shown in the top. In contrast, the joint angle values of $\mathbf{x}[35]$ are not affected by the torque but instead continuously increase over time.

the torque wrench slightly yields and consequently exerts no torque at the beginning of the tightening motion.

In contrast, $\boldsymbol{\omega}$ contains the information transfer received by the proprioceptors from the relative time. Here, the received information is almost completely contained in two proprioceptors which measure different joint angles. The joint angle $\mathbf{x}[35]$ with peak TE value ω_{35} is visualized in the bottom of Figure 6.2 right. As can be seen, the proprioception is not affected by different exerted torques and continuously increases over time. Hence, this proprioceptor is suitable for estimating the actual phase of the tightening motion. Furthermore, it can be observed that the angle values are slightly shifted (up to ten time steps) for different repetitions of the behavior. This confirms that the relative time is not sufficient to precisely describe the state of the behavior execution.

The particular amount of TE is then used to select the most relevant proprioceptors from the overall training data $\mathbf{X}$. Here, the proprioceptors with at least 90%[1] of the information transfer are selected and stored in separate matrices $\mathbf{X}_{\upsilon}$ and $\mathbf{X}_{\omega}$. In particular, $\mathbf{X}_{\upsilon}$ comprises torque related proprioceptors while $\mathbf{X}_{\omega}$ contains proprioceptive measurements which belong to the phase of the tightening motion. An interesting observation is that the estimates of the integrated force-torque firmware receives close to no information and are contained in neither of them. This can be attributed to an inaccurate configuration which confuses the estimates of the integrated firmware.

Next, two feature spaces are computed by performing PCA. Here, noisy proprioceptors are dampened by erasing the PCs with the lowest eigenvalues. In particular, the resulting Phase Feature Space (PFS) focuses on temporal correlations while the Torque Feature Space (TFS) emphasizes correlations to the torque. The projected trajectories

[1]The threshold is determined empirically but can be changed in order to adapt the computational demand.

of the corresponding training data are defined as

$$\begin{aligned} \mathbf{X}'_{\upsilon} &= \mathrm{TFS}(\mathbf{X}_{\upsilon}) \\ \mathbf{X}'_{\omega} &= \mathrm{PFS}(\mathbf{X}_{\omega}). \end{aligned} \tag{6.4}$$

Here, $\mathbf{X}'_{\upsilon} = [\mathbf{x}'_{\upsilon}[1], \ldots, \mathbf{x}'_{\upsilon}[m'_{\upsilon}]]$ contains $m'_{\upsilon} \leq m$ low-dimensional trajectories while $\mathbf{X}'_{\omega} = [\mathbf{x}'_{\omega}[1], \ldots, \mathbf{x}'_{\omega}[m'_{\omega}]]$ is composed of $m'_{\omega} \leq m$ trajectories. Hence, proprioceptors which are not influenced by the relative time are ignored during phase estimation while proprioceptors not related to the torque are ignored during the subsequent torque estimation. During the further procedure, the PFS and the corresponding projected training data $\mathbf{X}'_{\omega}$ are used for phase estimation. Here, knowledge about the actual phase helps to increase the accuracy of the subsequent torque estimation which is based on the TPS and the projection $\mathbf{X}'_{\upsilon}$.

6.1.3 Phase Estimation

The first step of this lazy V-BSPM implementation is to calculate the most similar phase for each of the given behavior examples. Due to small time shifts, occurring between multiple executions of a behavior, the relative time is not able to precisely estimate the actual state of the behavior execution. Instead, the actual phase is estimated from the PFS which is primarily depending on two of the robot's joint angles (as shown in Figure 6.2 left). Here, a sequence of the measured proprioception $\hat{\mathbf{X}} = [\hat{\mathbf{x}}(1); \ldots; \hat{\mathbf{x}}(t)]$ is projected into the PFS, where t is the number of samples or sequence length. The resulting low-dimensional trajectory

$$\hat{\mathbf{X}}'_{\omega} = \begin{bmatrix} \hat{\mathbf{x}}'_{\omega}(1) \\ \vdots \\ \hat{\mathbf{x}}'_{\omega}(t) \end{bmatrix} \tag{6.5}$$

is used to estimate the phase of the behavior execution. For this, the most similar sequence of $\hat{\mathbf{X}}'_{\omega}$ in $\mathbf{X}'_{\omega}$ needs to be detected.

Here, the Subsequence Dynamic Time Warping (SDTW) algorithm introduced in Section 5.2.4 can be adapted with regard to two beneficial properties of the acquired training data. (1) the proprioceptors which represent the temporal evolution of the behavior execution are independent from the k torque configurations (as shown for the joint angle proprioception in Figure 6.2 right). Therefore, only one instead of k behavior executions is sufficient to calculate a first estimate of the most similar sequence. (2) the interface of the UR5 robot provides an equidistant temporal interval. This reduces the number of possible warping paths to $n - t$. This reduces the SDTW optimization problem to finding the optimal path $\mathbf{p}^*_1 = (s^*_1, \ldots, e^*_1)^{\mathsf{T}}$, where $e^*_1 = s^*_1 + t - 1$ and s^*_1. Hence, only the start point s^*_1 of the most similar sequence between n training samples

and t runtime samples need to be calculated by

$$s^* = \underset{s\in[1:(n-t)]}{\operatorname{argmin}} \sum_{i=1}^{t} c(\hat{\mathbf{x}}'_\omega(i), \mathbf{x}'_\omega(s+i)). \tag{6.6}$$

As a result, the computational complexity of the SDTW algorithm is reduced from $\mathcal{O}(nkt)$ (see Equation 5.17 p. 81) to $\mathcal{O}((n-t)\cdot t)$ and no warping is required. A first estimate of the actual phase is determined by the relative time which is stored for the end point of the corresponding training data $w_{e_1^*}$.

So far, only the most similar phase for one behavior execution was taken into account. Due to the similarity of the phases, the search space for the $k-1$ other examples is reduced by applying a hill climbing approach. The starting point of the search space for the remaining training sets is at s_1^*. By applying Equation 6.6, the hill climbing method is searching the neighbors of s_1^* and stops when no smaller EUD can be found. Since $\mathbf{p}_1^*$ contains the global minimum for one behavior execution it can be assumed that, due to their similarity, $\mathbf{p}_2^*, \ldots, \mathbf{p}_k^*$ also contain global minima. Finally, the most similar phases for all of the recorded behavior examples are stored in $\mathbf{P}^* = [\mathbf{p}_1^*, \ldots, \mathbf{p}_k^*]$.

6.1.4 Torque Estimation

Based on the previous phase estimation procedure, the TFS is utilized to identify the most similar torque configuration which then is used to augment the robot's proprioception. For this, the sequence of the actually measured proprioception $\hat{\mathbf{X}}$ is projected into the TFS. The resulting low-dimensional trajectory has the following appearance

$$\hat{\mathbf{X}}'_\upsilon = \begin{bmatrix} \hat{\mathbf{x}}'_\upsilon(1) \\ \vdots \\ \hat{\mathbf{x}}'_\upsilon(t) \end{bmatrix}. \tag{6.7}$$

Usually, the actual projection $\hat{\mathbf{X}}'_\upsilon$ needs to be compared with all training samples $\mathbf{X}'_\upsilon$. The computational complexity of this procedure $\mathcal{O}((n-t)\cdot kt)$ (see Equation 5.17 p. 81) is time consuming. In order to decrease the number of possible comparisons, the procedure is reduced to a similarity analysis for the training samples with the most similar phases $\mathbf{P}^*$. The training sequence $\mathbf{x}'_\upsilon(\mathbf{p}_{l^*}^*)$ with the highest correspondence to the actual sequence $\hat{\mathbf{X}}'_\upsilon$ is calculated by

$$l^* = \underset{l\in[1:k]}{\operatorname{argmin}} \sum_{i=1}^{t} c(\hat{\mathbf{x}}'_\upsilon(i), \mathbf{x}'_\upsilon(\mathbf{p}_l^*)), \tag{6.8}$$

where $l^* \in [1 \ldots k]$. This reduces the computational complexity to $\mathcal{O}(kt)$. Finally, the torque value contained in the label of the training data $\mathbf{v}_{l^*}$ defines the V-BSPM output.

To generalize outputs for arbitrary inputs, these training sets are interpolated with an accuracy of 0.01 N m (as described in Section 5.2.2). For this, a two second inverse tightening motion was executed after finishing the original one. Here, the previously

introduced DMD interpolation technique is applied to extract the underlying dynamics of the cyclic movement pattern. This avoids a time consuming training phase but increases the number of behavior examples and consequently the computational demand. The resulting data set contains $k = 901$ equidistant examples for torques in between $(6.0\,\mathrm{N\,m}, \ldots, 15.0\,\mathrm{N\,m})$. This affects the computational demand of the torque estimation ($\mathcal{O}(kt)$) but has no influence on the phase estimation ($\mathcal{O}((n-t)\cdot t)$).

As the robot performs the movement repeatedly, the torque continuously increases. Hence, stopping the behavior execution at a certain level of torque requires a real-time approach. In order to ensure constant computational demand, a first estimate is calculated based on the non-interpolated behavior examples. The torque accuracy is then increased by searching the neighborhood of this estimate with the previously mentioned hill climbing method. This procedure has a maximal computation time which can be set to a value less than the data rate provided by the robot, e.g., 125 Hz for the UR5. This ensures the real-time capability of the V-BSPM and allows stopping the behavior execution when reaching a certain torque value.

6.1.5 Evaluation

First, the usage of TE with regard to the phase and torque estimation of the V-BSPM is evaluated. For this, three different proprioception groups are proposed for the PFS:

- $\mathrm{PFS}_{\mathrm{TE}}$ utilizes eight proprioceptors which receive at least 90% of the information transfer from the relative time.
- $\mathrm{PFS}_{\neg\mathrm{TE}}$ makes use of the remaining 97 proprioceptors which receive less than 10% of the information transfer.
- $\mathrm{PFS}_{\forall}$ does not apply TE and instead make use of all $m = 105$ proprioceptors for phase estimation.

The torque is randomly chosen and a sequence of low-dimensional projections for the last 25 proprioceptive measurements is provided. Figure 6.3 left shows the resulting phase estimation for the different proprioception groups. Here, only the $\mathrm{PFS}_{\mathrm{TE}}$ is able to estimate the actual phase accurately but is slightly shifted to the relative time. This can be explained by the different offsets at the beginning of a behavior execution. Instead of using the relative time, the $\mathrm{PFS}_{\mathrm{TE}}$ detects this offset and continuously estimates the correct phase. Furthermore, there is nearly no difference between the results of $\mathrm{PFS}_{\neg\mathrm{TE}}$ and $\mathrm{PFS}_{\forall}$ since suitable sensors are excluded or form only a small part. This shows that the $\mathrm{PFS}_{\mathrm{TE}}$ contains the most relevant sensors to estimate the actual phase of the behavior execution.

Next, the quality of the selected proprioceptors for V-BSPM torque estimation is evaluated by three different proprioception groups:

- $\mathrm{TFS}_{\mathrm{TE}}$ is generated from 18 proprioceptors which receive at least 90% of the information transfer from the exerted torque.

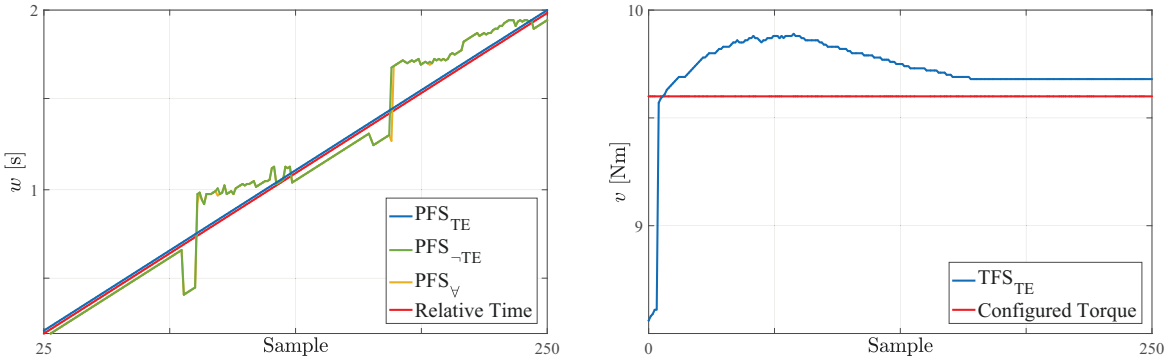


Fig. 6.3: The phase and subsequent torque estimation step of the V-BSPM. Left: V-BSPM phase estimation for different proprioception groups. PFS_{TE} is generated from the proprioceptors receiving 90% of the overall TE from the relative time while $\text{PFS}_{\neg\text{TE}}$ is generated from the remaining sensors and $\text{PFS}_{\forall}$ utilizes all sensors. As can be seen, only PFS_{TE} is able to accurately estimate the actual phase which is strongly correlated to the relative time. Right: V-BSPM torque estimation during real-time behavior execution. The screw nut is configured with 9.6 N m. The TFS_{TE} torque estimation results fail at the beginning because of the small sequence length of the recorded samples and the less accurate output of the phase estimation. The accuracy of the estimation directly depends to the size of the recorded real-time samples and gets closer to the configured torque after half a second.

- $\text{TFS}_{\neg\text{TE}}$ utilizes the remaining 87 proprioceptors which receive less than 10% of the information transfer.
- $\text{TFS}_{\forall}$ does not apply TE and instead make use of all $m = 105$ proprioceptors.

Furthermore, the training data which was executed for a torque of 8.0 N m is utilized as validation data $\mathbf{Y}$. To prove the robustness of the V-BSPM torque estimation, $\mathbf{Y}$ is projected into the different TFSs $\mathbf{Y}'_v$ and disturbed. In particular, each low-dimensional trajectory is disturbed by white noise of the following signal-to-noise ratio

$$SNR = \frac{\sigma^2_{\mathbf{Y}'_v}}{\sigma^2_{noise}}, \tag{6.9}$$

where σ is the standard deviation calculated from ten repetitions of the tightening motion with 8.0 N m. Table 6.1 summarizes the resulting MAEs for different levels of noise. Similar to phase estimation, the highest accuracy is achieved by utilizing TFS_{TE} while $\text{TFS}_{\neg\text{TE}}$ and $\text{TFS}_{\forall}$ result in incorrect estimations. This demonstrates that the proprioceptors which receive the highest amount of information transfer are a suitable choice for accurate torque estimation of the V-BSPM.

Finally, the V-BSPM real-time accuracy for an interpolated torque configuration is evaluated. More precisely, the robot performs the tightening of the nut which was configured with a torque of 9.6 N m. In order to avoid an additional tightening, the goal is to identify this torque as fast as possible. For this, each sample is projected into the PFS_{TE} and appended to a sequence of the previous projections. The sequence length of this low-dimensional trajectory is set to 25 which, as evaluated in the previous section, results in accurate phase estimation results. As illustrated in Figure 6.3 left, before this trajectory contains 25 elements, no phase estimation is solved. Instead, the relative time is utilized to approximate the actual phase at the beginning of the

Tab. 6.1: The MAEs in N m resulting from different proprioception groups and signal-to-noise ratios. The error signal is generated utilizing white noise.

	$SNR = 20$	$SNR = 10$	$SNR = 5$	$SNR = 2.5$	$SNR = 1$
TFS_{TE}	**0.0**	**0.0**	**0.0**	**0.01**	**0.07**
$\text{TFS}_{\neg\text{TE}}$	1.05	1.34	1.42	1.74	2.51
$\text{TFS}_{\forall}$	0.53	0.89	1.03	1.18	1.75

behavior execution.

As illustrated in Figure 6.3 right, this negatively influences the subsequent torque estimation procedure. Here, the sequence length t of the corresponding TFS_{TE} is not limited and continuously increases. Consequently, also the computational costs increase but still require less than 8 ms for a maximum sequence length of 250. After containing 25 time steps, the estimated torque is equivalent to the configured one. Taking a closer look at the results, the offset after half a second is 0.3 N m and after two seconds still about 0.1 N m. The length of the torque wrench from the original TCP is exactly 0.23 m. This results in an accuracy of $\frac{0.3\,\text{N m}}{0.23\,\text{m}} = 1.3\,\text{N}$ after half a second and $\frac{0.1\,\text{N m}}{0.23\,\text{m}} = 0.43\,\text{N}$ at the end of the behavior execution. Thus, the proposed lazy V-BSPM augments the robot's proprioception with accurate torque estimates which outperform the precision of the UR5's integrated firmware ($\pm 25\,\text{N}$) by an order of magnitude.

In contrast to this experiment, the configured torque value is usually not known to the V-BSPM during runtime. Without a comparison between ground truth and estimated torque value extrinsic perturbations remain hidden and obfuscate the estimation results. Hence, the user has to guarantee a regular runtime execution of the behavior to ensure the applicability of the V-BSPM. By specifying adequate reaction rules, the augmented proprioception can then be utilized for behavior adaptation. For example, to avoid an additional tightening of the screw nut, the robot may stop the behavior execution or perform a reverse motion to configure the screw nut to the initial value.

6.1.6 Conclusion

In this section, a V-BSPM was applied to estimate the exerted torque during a manipulation task where an overview is given in Figure 6.4. Here, training examples contain proprioceptive patterns which are recorded from regular behavior executions for varying contexts. These patterns represent the context specific intrinsics and are assigned with corresponding numeric labels in a torque range from $6\,\text{N m} - 15\,\text{N m}$. For the introduced tool usage scenario, the context is defined by the relative execution time and different torque configurations. Here, TE and PCA are utilized to extract the most relevant features for phase and torque estimation from the overall set of proprioceptors. In contrast to the I-BSPM introduced in the previous section, TE is directly applied to the raw proprioceptive measurements. The proprioceptors which receive the highest amount of information transfer are selected for PCA. The resulting low-dimensional PFS and TFS are strongly correlated to the corresponding time/torque

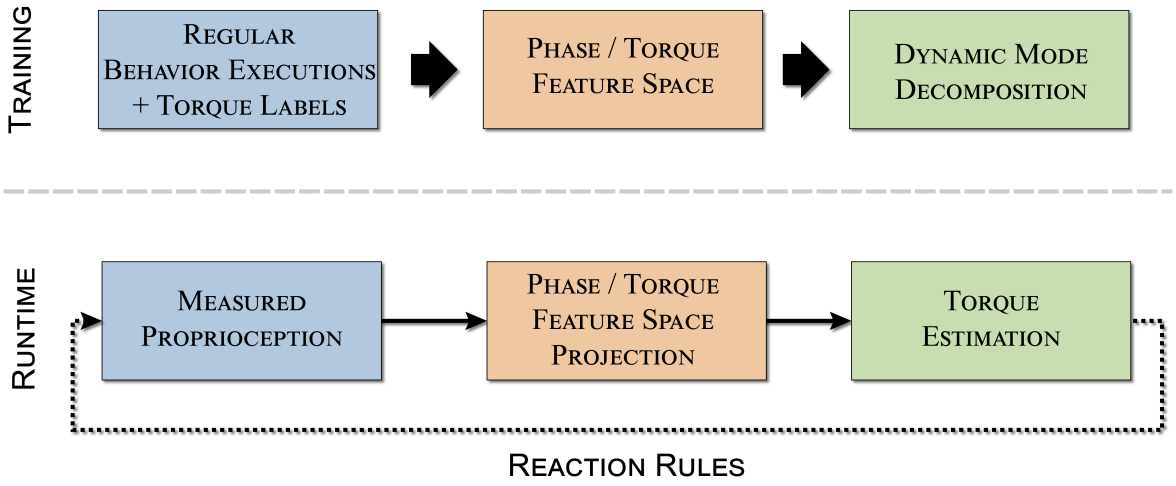


Fig. 6.4: Overview of the presented lazy V-BSPM approach with regard to the methodology introduced in Section 4.3. Regular behavior examples containing the proprioception together with context specific torque labels are utilized to build a PFS and a TFS. Here, DMD is applied to interpolate unknown proprioceptive patterns for arbitrary torque configurations. At runtime, a sequence of the measured proprioception is projected into the PFS and mapped to the nearest training example by utilizing an adapted SDTW algorithm. Here, the most similar phase is utilized to increase the efficiency of the subsequent TFS mapping procedure. The torque which was configured for the most similar training example is then used as an estimate of the actual torque. This estimate augments the robot's proprioception and can be used to adapt the behavior execution by specific reaction rules.

specific selection. These feature spaces are the central point of the presented V-BSPM.

At runtime, a mapping of the measured proprioception to the most similar interpolated training examples is achieved by utilizing an adapted SDTW algorithm. The adapted algorithm reduces the computational complexity of the classical SDTW method (see p. 95). This ensures the real-time applicability of the given tool usage task. In particular, the mapping is computed in two subsequent steps: (1) phase estimation which reduces the search space of (2) torque estimation. The label of the training example with the most similar proprioceptive pattern as the actually measured one is used as output of the V-BSPM. Here, a torque accuracy of 0.3 N m is achieved after half a second. This compares to the capabilities of an experienced mechanic.

To this end, a drawback of the proposed V-BSPM are the dependencies between the amount of training data and the required computational effort. Here, with increasing data size also the computation time increases. This leads to a delayed reaction and consequently negatively impacts the robot's safety. Restricting the procedure to a maximum computation time may result in a constant reaction time but decreases the estimation accuracy and also has a negative impact on safety. Solving this by scaling the provided computational power contradicts the motivation of the proposed approach. Hence, as for most lazy learning techniques the amount of training data adversely affects the scalability of the proposed implementation. In the following, a neural network is used to implement an eager V-BSPM. Here, a main advantage is the constantly low reaction time which is independent from the amount of training data.

Fig. 6.5: A mobile robot platform equipped with a UR5 robotic arm and a 3-fingered gripper. The robot is deployed in an underground mine in order to conduct various monitoring and exploration tasks. Here, one application is to draw water samples by utilizing a self-contained water extraction station. Hence, the fill level of the sample vessel cannot be obtained from the station directly. Furthermore, a FT sensor is not applicable because of problems related to roughness, load capacity and accuracy. Instead, a V-BSPM is utilized to augment the robot's proprioception with the ability to classify the weight and consequently the fill level of the sample vessel.

6.2 Deep Learning V-BSPM

This application aims to learn a neural network based V-BSPM for a weight classification task of the UR5 robotic arm. More precisely, as shown in Figure 6.5, the UR5 robot is mounted on a mobile robot platform which is used for mapping and monitoring in underground mines [125][126]. One goal of this platform is to collect water samples from its environment [127]. For this, the robot is equipped with a low-cost water extraction station [128] which has a sample vessel with a maximum fill volume of 250 ml but no capacity sensor. When the robot is requested to take a water sample it grasps the station from a fixed docking position and puts it on the ground. The extraction process is automatically started when the station is getting in contact with liquid. Due to various reasons [128], the station is self-contained and therefore cannot communicate with the robot. Hence, the extraction process is stopped after a predefined duration of 25 seconds which was hopefully long enough to extract a relevant amount of water. Next, the robot performs a five second behavior which lifts the station 50 cm about the ground. Consequently, there is no pre-built solution to classify the fill level of the sample vessel (empty, half-full, full).

One could argue that the fill level corresponds to the station's weight which can be determined by utilizing a FT sensor. Because of the following reasons this is not feasible for the applications of the proposed robot platform:

- Roughness: A major challenge of the robot platform is its application under rough environmental conditions. For the examined underground mine all electronic

parts need to provide protection against dust and spray water (IP67 [1]). Hence, an adequate FT sensor requires the same level of protection. This can be achieved by a costly external cover which adds additional weight to the robot.

- Load capacity: As specified by the manufacturer, the load capacity of the UR5 is limited to 5 kg. The utilized gripper together with the filled water station already requires 4.1 kg. The FT150 adds an additional weight of 0.65 kg. Together with the protective cover and power cables this result in a marginal overall weight.

- Accuracy: The manufacturer of the FT150 specifies an accuracy of ± 0.5 N which is a common value for state-of-the-art FT sensors. For the elaborated setup, this corresponds approximately to ± 50 ml which as will be shown in the later data acquisition step, is not sufficient.

Instead, a V-BSPM is applied to augment the robot's proprioception with the ability to estimate the weight of the station, i.e., to classify its fill level.

For this, the robot performs regular lifting behaviors under different contexts while recording the raw sensor readings of the UR5 robot. By utilizing information-theoretic measures, a subset of proprioceptors is selected from this training data. The remaining proprioceptors are investigated by an ANN which utilizes a state-of-the-art deep learning technique to estimate the weight of the water extraction station and consequently the fill level of the sample vessel. General advantages of ANNs are their scalability to the amount of training data, their ability to estimate values in constant time and the generalization of outputs for arbitrary inputs. An detailed description of ANNs and their functionality is given in Section 3.2. In the following, each step of this neural network based V-BSPM implementation is explained in more detail.

6.2.1 Data Acquisition: Behavior Examples with Context Labels

The proposed classification task does not rely on additional hardware and therefore focuses on learning a V-BSPM from manually labeled data. Here, the robot needs to place the water extraction station (shown in Figure 6.6 left) which was developed by Gebel [128]. After a predefined extraction time, the robot lifts the station where the basic idea is to distinguish between an empty, half-full and full sample vessel for arbitrary locations within the robot's workspace. The workspace of the robot is illustrated in Figure 6.6 right (green area) where the markers describe several predefined lifting positions. Similar to a human guessing motion, the robot gathers experience by performing the same behavior under varying well-known conditions. More precisely, for each of the illustrated positions, the robot performs a behavior which lifts the station 50 cm within exactly five seconds resulting in 625 equidistant measurements.

Hence, one recording for the blue positions results in $n_X = 9375$ measurements where each $\mathbf{x} = (x_1, \ldots, x_m)$ contains the UR5 robot's $m = 105$ proprioceptors. The resulting

[1] IEC standard 60529

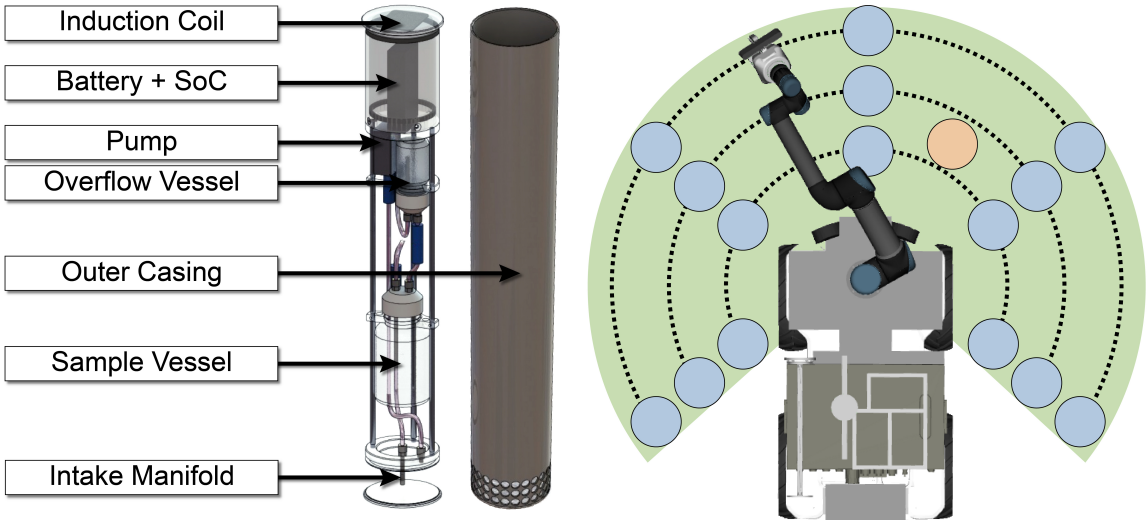


Fig. 6.6: A water extraction station is utilized by the mobile robot platform. Left: The self-contained water extraction station is constructed to automatically start and stop the extraction process but provides no communication interface. adopted from [128, p. 15–17] Right: A lifting behavior is performed at several positions inside the robot's workspace which is approximately limited by the green area. Blue markers describe lifting positions which are utilized for the training data set. The red one is not contained in the training data and therefore is suitable for validation purposes.

training data for a constant fill level y has the following appearance

$$\mathbf{X}_1 = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{n_X} \end{bmatrix}, \mathbf{y}_1 = \begin{pmatrix} y_1 = y \\ \vdots \\ y_{n_X} = y \end{pmatrix}. \tag{6.10}$$

As stated above, the task is to enable the robot to distinguish between an empty, partially-filled and filled sample vessel. Hence, the lifting behavior is recorded for $k = 3$ manually configured fill levels $\langle 0, 125, 250 \rangle$ [ml] on the blue positions. The total training data is then defined by

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_k \end{bmatrix}, \mathbf{y} = \begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_k \end{pmatrix}. \tag{6.11}$$

For validation, the sample vessel was filled with 100 ml and lifted four times at the red position. The corresponding recording contains $n_V = 2500$ measurements and is referred to as validation data $\mathbf{V}$. It is notable that the lifting position and the weight of $\mathbf{V}$ are not contained in $\mathbf{X}$, what makes it suitable for evaluating the later learning process. Finally, the measured proprioception in $\mathbf{X}$ and $\mathbf{V}$ is preprocessed by means of normalization (z-score) and quantization (a four bit binning estimator). This increases the efficiency of the subsequent correlation and learning procedure.

As stated above, the manual labels are required since the accuracy of state-of-the-art FT sensors is not sufficient to solve this task. To show this, the measurements of the FT150 are recorded during the conducted acquisition process of the training examples.

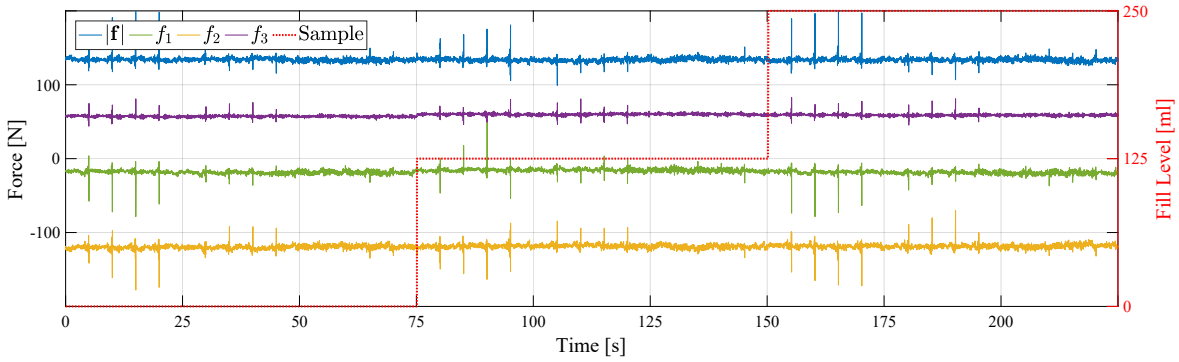


Fig. 6.7: The mobile robot platform utilizes a FT150 which is attached to the TCP of the UR5 robotic arm to measure the three-dimensional forces $\mathbf{f} = (f_1, f_2, f_3)$ and the corresponding overall strength $|\mathbf{f}|$. The recorded training data $\mathbf{X}$ is concatenated where 0 s-75 s represent empty, 75 s-150 s half-full and 150 s-225 s full lifting behaviors. The force measurements are disturbed by the inertia of the water and the acceleration of the robot. Hence, the fill level is not evident from these measurements and the FT150 is not applicable to solve the corresponding classification task.

Figure 6.7 shows the acquired three-dimensional forces $\mathbf{f} = (f_1, f_2, f_3)$ and their overall strength $|\mathbf{f}|$, compared to the actual fill level of the sample vessel. Here, the training examples are concatenated as follows:

- 0 s-75 s: Empty sample vessel with a fill level of 0 ml.
- 75 s-150 s: Half-full sample vessel with a fill level of 125 ml.
- 150 s-225 s: Full sample vessel with a fill level of 250 ml.

Hence, the weight applied to the TCP $|\mathbf{f}|$ should highlight the difference between these examples. As can be seen, the fill level is not evident from the measurements of the FT150. This is due to the fact that the inertia of the water together with the acceleration of the robot causes additional noise. One could argue that with some further effort the accuracy of the measurements can be increased. For example, the robot can stop the behavior execution for a few seconds and calculate the mean value of the forces. However, this additional routine needs to be implemented by the user, requires interrupting the behavior execution and is still no guarantee for precise measurements. Hence, without any further effort, the accuracy of this state-of-the-art FT sensor is not sufficient to solve the proposed task.

Instead of using a FT sensor, a virtual force sensor is learned from the acquired behavior examples. To this end, the fill level $\mathbf{y}$ is utilized as learning target of the V-BSPM while the proprioceptive measurements $\mathbf{X}$ are used as input. After finishing a runtime extraction process, the robot has to put the station back to its docking position. For this, the station is lifted by a five second behavior which is similar to the ones contained acquired for training. This has the advantage that no additional routine need to be specified by the user. The resulting continuous proprioceptive measurements are then used for weight estimation which allows classifying the fill level of the station. As will be shown, the V-BSPM already generates a precise decision after less than half a second of lifting.

Due to the small number of behavior examples, spurious correlations within the proprioceptors and the fill level can result in an overfitting of the learning procedure. In more detail, the spurious correlated information allows the ANN to learn an precise mapping of the training examples while losing the ability to generalize outputs for arbitrary inputs (see p. 43). To avoid this, the learning procedure monitors the generalizability of the V-BSPM by utilizing the validation data $\mathbf{V}$. This allows stopping learning when overfitting begins. Furthermore, following the BSPM approach (see p. 58), information-theoretic measures are applied to erase less relevant and redundant proprioceptors.

6.2.2 Iterative Proprioceptor Selection

In the previous lazy V-BSPM approach, TE was utilized to select a subset of proprioceptors which receive information transfer from the target. For this, the proprioceptors with the highest information transfer were directly selected without taking into account each other, e.g., two or more proprioceptors with a high TE may contain similar information and therefore are in parts or completely redundant. The previous task benefits from such redundancies since one of the proprioceptors may be temporarily influenced by noise. Hence, redundancy in proprioceptive information helps to compensate the disturbed proprioceptor which ensures meaningful results. However, such redundancy is less relevant for the present classification task since only one decision has to be made after finishing the behavior execution. For this, a minimal subset of proprioceptors which contain the maximum of mutual information with the learning target, e.g., the fill level, is selected by utilizing Conditional Mutual Information (CMI) and Multivariate Mutual Information (MMI) (Section 3.1.2).

The basic idea is to iteratively select a subset of proprioceptors $\tilde{\mathbf{X}}$ with relevant information about the actual state of the fill level $\mathbf{y}$. On the one hand, the CMI $I(\mathbf{y}; \mathbf{x}[i] - \delta_j | \tilde{\mathbf{X}})$ determines the information shared between the fill level $\mathbf{y}$ and the ith proprioceptor $\mathbf{x}[i]$ which is not already contained in the selection $\tilde{\mathbf{X}}$. Here, $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_j)$ defines multiple possible time lags which allows to recognize time delayed dependencies. On the other hand, the MMI $I(\mathbf{y}; \mathbf{x}[i] - \delta_j; \tilde{\mathbf{X}})$ quantifies the amount of information gained compared to the already contained information. Consequently, a positive MMI indicates a predominant portion of redundancy while negative values imply synergistic effects (see Equation 3.10 p. 31). The resulting iterative selection procedure was already introduced in Algorithm 1 (see p. 58).

As it turns out, the proprioceptors selected with MMI contain less redundancy, while proprioceptors selected with CMI maximize the mutual information. For this reason, the selection procedure usually provides better results when utilizing CMI rather than MMI. Then, the CMI and a similar MMI algorithm are applied for different time delays $\boldsymbol{\delta} = (0, \ldots, 10)$. Figure 6.8 shows the increasing ratio of mutual information and Shannon entropy $I(\tilde{\mathbf{X}}; \mathbf{y}) \cdot H(\mathbf{y})^{-1}$. Both measurements require ten proprioceptors in the subset $\tilde{\mathbf{X}}$ to gather about 99% of mutual information with the fill level $\mathbf{y}$. As expected, due to ignoring the amount of redundancy, the information gain when utilizing

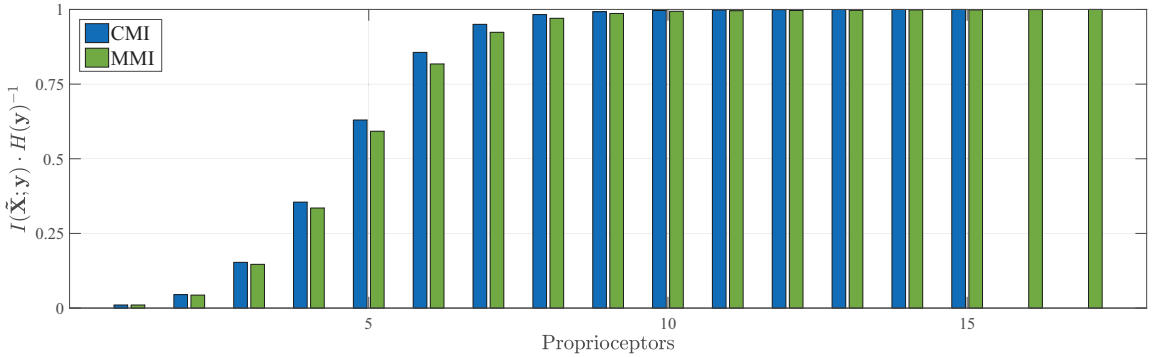


Fig. 6.8: Utilizing the proposed iterative proprioceptor selection procedure (Algorithm 1 p. 58), CMI and MMI are applied to the training data. The CMI selection grows faster and requires less proprioceptors $\tilde{\mathbf{X}}$ to obtain 100% mutual information with the fill level $\mathbf{y}$. This is due to the fact that MMI relies on the relation between redundancy and synergy while CMI focuses on the overall mutual information.

CMI is growing faster than MMI and requires only 15 instead of 17 proprioceptors to completely obtain $\mathbf{y}$. Consequently, less proprioceptors are required to completely obtain the actual state of the fill level. This makes CMI to the method of choice for the proposed task. In the following, the ten proprioceptors with the highest combined CMI $\tilde{\mathbf{X}}_{\text{CMI}_{10}}$ are utilized for training while $\tilde{\mathbf{V}}_{\text{CMI}_{10}}$ is used for validation purposes.

6.2.3 Deep Learning Classifier

In this section, a deep neural network classifier is learned by the V-BSPM approach from the previously selected subset of proprioceptors $\tilde{\mathbf{X}}_{\text{CMI}_{10}}$. More precisely, an ANN architecture is applied which (1) implements an indefinitely memory and (2) provides a categorical probability distribution as output.

The first is realized by implementing Long Short-Term Memory (LSTM) blocks, which was depicted in Section 3.2.4. In contrast to classical recurrent architectures, these blocks provide an indefinite memory. More precisely, these blocks have the ability to remember information for an infinite delay. Each repetition of the lifting behavior is represented by time series data of 625 equidistant measurements. For each time step, the classifier provides an estimate of the station's fill level. Here, the usage of LSTM blocks allows remembering the past proprioception for the complete sequence of time series data. As a result, the classifier gets more confident with each additional proprioceptive measurements of the behavior execution. This further has the advantage that spontaneous noise is filtered automatically.

The second is achieved by utilizing a *softmax activation function* [129] inside the output neurons of the corresponding network. This function squashes inputs $\mathbf{inp} = (inp_1, \ldots, inp_r)$ to the same size of outputs $\mathbf{out} = (out_1, \ldots, out_r)$ in the range between $(0, 1)$ by

$$out_j = f_{softmax}(\mathbf{inp}, j) = \frac{e^{inp_j}}{\sum_{i=1}^{r} e^{inp_i}}, j \in [1 \ldots r]. \tag{6.12}$$

In contrast to the usage of a sigmoidal or linear activation function, the sum of the soft-

max outputs $\sum_{i=1}^{r} out_i = 1$ and therefore **out** is equivalent to a categorical probability distribution.

To provide a probability based output, the fill levels $y \in \{0\,\text{ml}, 125\,\text{ml}, 250\,\text{ml}\}$ are categorized in classes $y \in \{(1,0,0),(0,1,0),(0,0,1)\}$. The indefinite memory is then implemented by an LSTM network architecture which is constructed of:

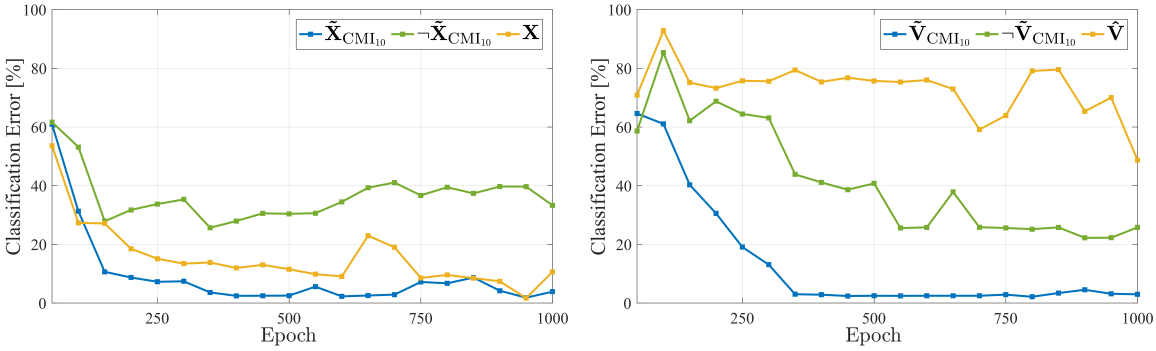
- Input layer: for each time step, the selected proprioceptors $\tilde{\mathbf{X}}_{\text{CMI}_{10}}$ are utilized as input neurons. These ten neurons are fully connected to the cell inputs and gate units contained in the hidden layer what results in 400 connections.
- Hidden layer: there is one hidden layer which is composed of ten LSTM blocks. In turn, each block contains its gate units and exactly one cell. The cell outputs are transmitted to the output layer but are also fed back to all cell inputs and gate units what results in 430 additional connections.
- Output layer: the number of output neurons is equivalent to the dimensionality of the fill level classes. Hence, three output neurons are used as an estimate of the actual class. Finally, these neurons utilize a softmax activation function and therefore return a categorical probability distribution.

The resulting network consists of 830 connections and is referred to as LSTM_{CMI}. Furthermore, each connection contains a bias neuron which doubles the number of weighted connections to 1720. Due to this large amount of connections, a visual representation of LSTM_{CMI} is omitted.

Training the proposed LSTM_{CMI} requires a deep learning process which adapts the connection weights with regard to the training data. More precisely, the 45 behavior examples $\tilde{\mathbf{X}}_{\text{CMI}_{10}}$ are separated into sequences with a length of 625 equidistant measurements. Here, each sequence is iteratively fed into the input layer of LSTM_{CMI} where softmax function returns a discrete set of 625×3 elements. This set describes the categorical probability distribution for the different fill levels and is evaluated by utilizing the Cross Entropy Error (CEE) (see Equation 3.28 p. 43) The weights are then adapted by utilizing offline learning and the backpropagation procedure introduced in Section 3.2.3. This process is repeated for various epochs and is usually stopped when overfitting occurs or the classification accuracy is sufficient.

To this end, the class with the highest probability is utilized as the estimate of the actual class. An estimate which is not equivalent to the correct class contained in $\mathbf{y}$ is interpreted as classification error. The percentage of classification errors with regard to the learning epoch is shown in Figure 6.9(a). Here, the ten most beneficial proprioceptors $\tilde{\mathbf{X}}_{\text{CMI}_{10}}$ are compared to all sensors $\mathbf{X}$ and to the less beneficial $\neg\tilde{\mathbf{X}}_{\text{CMI}_{10}}$. As can be seen, the proposed selection approach outperforms the usage of the less beneficial sensors. This is due to the fact that most of the useful information is discarded from $\neg\tilde{\mathbf{X}}_{\text{CMI}_{10}}$. Similar results are achieved by the usage of all sensors, but the increased size of input neurons also requires considerably more computational effort.

Another drawback of utilizing all sensors is shown in Figure 6.9(b). Here, the validation data $\mathbf{V}$ is utilized to monitor the models generalizability for arbitrary inputs. As



(a) Training (b) Validation

Fig. 6.9: The evolution of classification errors during deep learning of LSTM networks with measurements of the most beneficial, less beneficial and all proprioceptors. Left: $\tilde{\mathbf{X}}_{\mathrm{CMI}_{10}}$ contain 99% information about the fill level $\mathbf{y}$ and therefore the network learns to accurately map the corresponding class. The remaining proprioceptors $\neg\tilde{\mathbf{X}}_{\mathrm{CMI}_{10}}$ contain minor information about the fill level and result in frequent classification errors. Utilizing all proprioceptors $\mathbf{X}$ requires the network to extract dependencies between the proprioceptors and the fill level itself. The resulting estimates are similar to the usage of $\tilde{\mathbf{X}}_{\mathrm{CMI}_{10}}$ but require considerably more computational effort. Right: After 300 learning epochs the most beneficial proprioceptors accurately estimate the correct class of the validation data $\tilde{\mathbf{V}}_{\mathrm{CMI}_{10}}$. This outperforms the usage of the less beneficial proprioceptors $\neg\tilde{\mathbf{V}}_{\mathrm{CMI}_{10}}$. Furthermore, the network trained with all proprioceptors is utilizing spurious correlations which are only contained in the training data and cannot generalize an accurate classification for the validation data $\mathbf{V}$.

stated above, the position and fill level (100 ml) of $\mathbf{V}$ are not contained in the training data $\mathbf{X}$. Hence, an accurate classifier should assign the highest probability to the class which is equivalent to the closest fill level $(0, 1, 0) \equiv 125\,\mathrm{ml}$. The classification errors for all sensors $\mathbf{X}$ reflect a major problem of learning with few behavior examples. More precisely, the network utilizes dependencies between the proprioceptors and the fill level which are only correct within the limited size of training data. The corresponding spurious correlations are not contained in $\mathbf{V}$ what therefore distracts the estimates. The resulting classification error is even higher than for the less beneficial proprioceptors $\neg\tilde{\mathbf{V}}_{\mathrm{CMI}_{10}}$. Latter does only contain less relevant information and also results in a high classification error. In contrast to that, $\tilde{\mathbf{V}}_{\mathrm{CMI}_{10}}$ achieves good results after 300 epochs and the most accurate estimate after 700 epochs. Here, repeating the learning process for 1000 epochs is slightly deteriorating the classifier's generalizability. As a result, the $\mathrm{LSTM}_{\mathrm{CMI}}$ is adjusted with the deep network's weight configuration learned after 700 epochs.

6.2.4 Fill Level Classification

At runtime, the measured proprioception is preprocessed in the same manner as the training data and subsequently processed by the classifier $\mathrm{LSTM}_{\mathrm{CMI}}$. Here, the network's processing time for one runtime measurement is 1.2 ms with a standard deviation of 0.2 ms. This is far less than the 8 ms provided by the interface of the UR5 robot and ensures the real-time applicability of the V-BSPM approach.

Each behavior execution results in a sequence containing 625 proprioceptive mea-

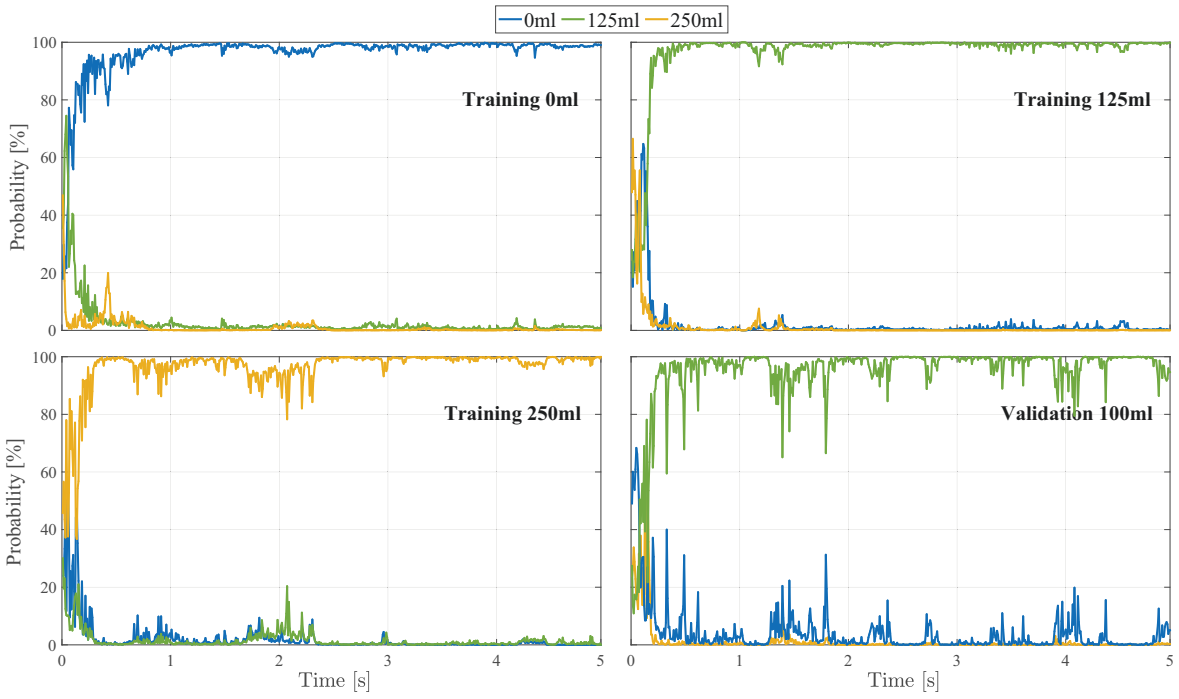


Fig. 6.10: The LSTM_{CMI} estimates the fill level by transforming each of the 625 inputs to a categorical probability distribution. Here, the output generated for four behavior executions with varying fill levels are shown. The classifier is getting more confident with each additional input and a correct decision can be usually met after half a second. As shown for the validation sequence, LSTM_{CMI} is also generalizing adequate estimates for arbitrary inputs.

surements of time series data. This sequence is used as the network's input which returns a categorical probability distribution. Due to its indefinite memory, the classifier is getting more confident about the fill level with each additional input. Figure 6.10 shows this fact for three different sequences contained in the training data $\tilde{\mathbf{X}}_{\text{CMI}_{10}}$ and one sequence of the validation data $\tilde{\mathbf{V}}_{\text{CMI}_{10}}$. Here, a high certainty about the fill level is achieved after less than half a second. Hence, the lifting behavior does not need to be finished to meet an accurate decision.

The most probable class is then used to adapt the extraction process. For the proposed task three classes are sufficient to enable the robot to distinguish between a correct, partially successful and incorrect extraction process. More precisely, the robot can utilize its augmented proprioception to implement a set of reaction rules:

- $0\,\text{ml} \equiv (1,0,0)$: in case of an incorrect extraction it is assumed that that the station was not in contact with water and therefore the robot has to adapt its lifting position.
- $125\,\text{ml} \equiv (0,1,0)$: a partially filled station indicates that an adequate position was selected but the process need to be repeated for a longer period.
- $250\,\text{ml} \equiv (0,0,1)$: in case of a successful extraction process the station is returned to its docking position.

In the following several experiments are conducted to further evaluate the accuracy, parameter configuration and robustness of the proposed V-BSPM classifier.

6.2.5 Evaluation

One could argue that a classical recurrent structure together with a softmax layer may be sufficient to solve the proposed task. In the following, the advantages of LSTM over classical Recurrent Neural Network (RNN) is demonstrated for the given classification task. For this a recurrent architecture is constructed with regard to the definitions in Section 3.2.2.

To ensure a high comparability, this RNN is constructed with a similar complexity as the LSTM_{CMI}. In particular, the input and output layer are equivalent to that contained in LSTM_{CMI}. Furthermore, the network contains two hidden layers which are fully connected with each other. The first hidden layer consists of ten and the second of five neurons where all neurons utilize a sigmoid activation function. Both layer outputs are fed back as input to itself with a maximal delay of ten. This enables the network to remember its internal activation which corresponds to the received inputs of proprioceptive measurements. In contrast to the indefinite memory of LSTM blocks, this limitation dampens the vanishing gradient problem which was extensively discussed in Section 3.2.3. Finally, the cell outputs of the second hidden layer are transmitted to the output layer. The resulting network contains an overall of 2830 weights and is referred to as REC_{CMI}.

Backpropagation is utilized to adapt the connection weights by an offline learning procedure on the training data $\tilde{\mathbf{X}}_{\text{CMI}_{10}}$. Hence, the complexity and the computational effort spent for learning REC_{CMI} is similar to that required for LSTM_{CMI}. The learning process is repeated from ten initial weight configurations for a maximum of 1000 epochs. The weight configuration which achieves the highest classification accuracy is then utilized as the final REC_{CMI}. Figure 6.11 compares the classification capabilities of both network architectures with regard to the training data $\tilde{\mathbf{X}}_{\text{CMI}_{10}}$. Here, REC_{CMI} results in 32.45% wrongly assigned classes while LSTM_{CMI} has an overall classification error of 2.53%. The less accurate results of the REC_{CMI} can be explained by the vanishing gradient problem. In particular, the network is unfolded in time what results in an increased number of hidden layers. As a result, backpropagation achieves only a low learning curve which requires much more epochs for an adequate adaptation of the network weights. In contrast to that, the usage of LSTM blocks almost eliminates the vanishing gradient problem and enables LSTM_{CMI} to remember the proprioception for the complete execution of the behavior. The remaining classification errors are, if at all, periodically contained at the beginning of a sequence. This corresponds to the effects observed in Figure 6.10. Consequently, LSTM_{CMI} is well suited for the proposed classification task while REC_{CMI} is confronted with the vanishing gradient problem.

A further difference to the previous lazy V-BSPM implementation is the iterative usage of CMI rather than TE. As stated above, TE does not take into account conditional information between multiple proprioceptors and therefore contain redundant

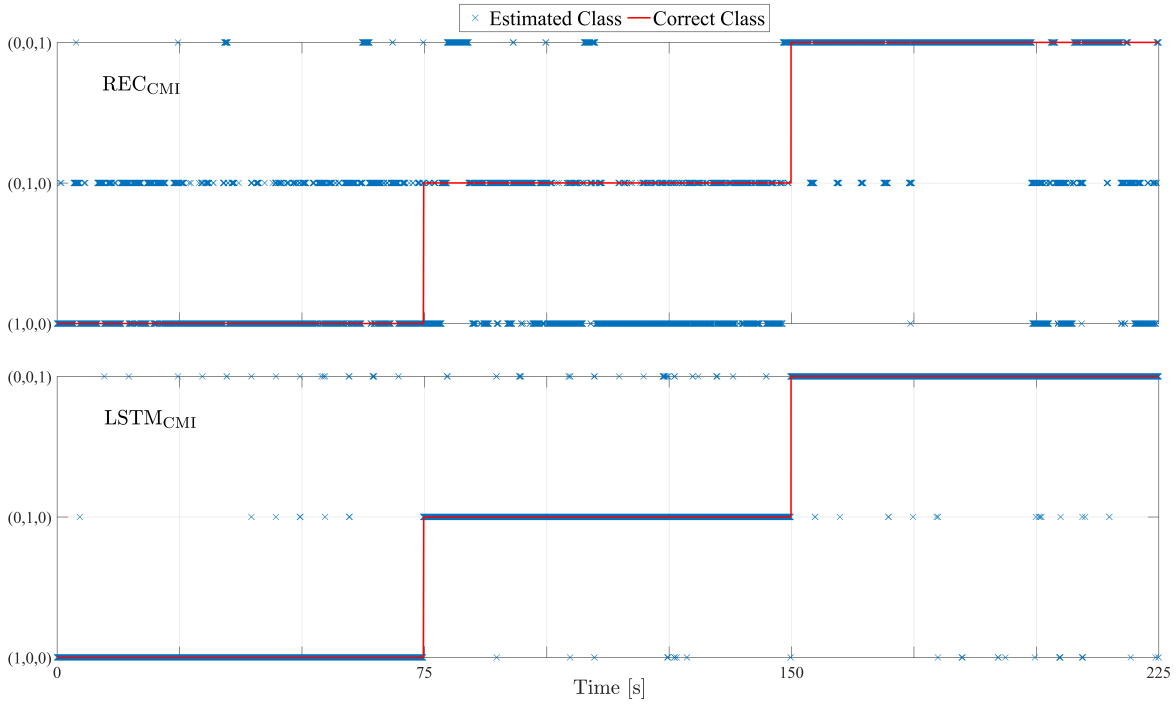


Fig. 6.11: The classification capabilities of REC_{CMI} and LSTM_{CMI} with regard to the training data. After a maximum of thousand epochs, REC_{CMI} wrongly assigns 32.45% of the overall inputs while LSTM_{CMI} has a classification error of 2.53%. For LSTM_{CMI} the remaining errors are periodically contained at the beginning of the behavior execution. This corresponds to the effects observed in Figure 6.10.

Tab. 6.2: The mean and best classification errors resulting from different selection schemes and network architectures. The particular networks were trained ten times from randomly chosen initial weights configurations where the learning process was repeated for a maximum of thousand epochs.

	REC_{CMI}	REC_{TE}	LSTM_{CMI}	LSTM_{TE}
Mean	40.32	35.83	**2.73**	15.01
Lowest	32.45	22.16	**1.46**	6.23

parts of information. To prove this assumption, TE is applied to detect a subset of ten proprioceptors which receive the highest information transfer from the fill level $\mathbf{y}$. These proprioceptors are selected from the training data $\tilde{\mathbf{X}}_{\text{TE}_{10}}$ and utilized for learning an LSTM and a classical recurrent network classifier for a maximum of thousand epochs. The corresponding networks are referred to as LSTM_{TE} and REC_{TE}. Table 6.2 compares the achieved classification capabilities for the proposed network architectures. More precisely, the mean and best classification errors for a set of ten randomly initialized weight configurations are computed. In fact, REC_{TE} achieves more accurate results than REC_{CMI} but is still confronted with the vanishing gradient problem. As expected, the indefinite memory of the LSTM networks overcomes this problem and perform much better. Here, the best results are achieved by LSTM_{CMI} which is learned by $\tilde{\mathbf{X}}_{\text{CMI}_{10}}$ rather than $\tilde{\mathbf{X}}_{\text{TE}_{10}}$. Hence, the proposed iterative selection procedure is beneficial for the usage of LSTM blocks while a redundant selection is applicable for the

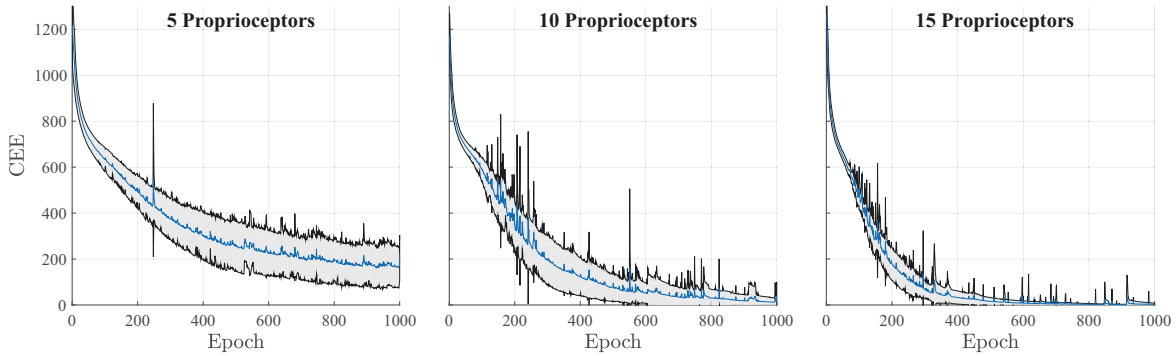


Fig. 6.12: The number of proprioceptors utilized as input variables are influencing the mean learning curve (blue) and standard deviation (gray area) of the particular LSTM. Left: The fill level share 63% information with a selection of five proprioceptors. Therefore, the learning curve converges early. Middle: Ten proprioceptors contain more than 99% information and therefore learn an adequate classifier in less than thousand epochs. Right: Utilizing fifteen proprioceptors with 100% mutual information results in the fastest learning curve but has an increased risk of containing less relevant information such as redundancy or add spurious correlations.

short-term memory of classical RNNs.

As stated above, the proprioception $\tilde{\mathbf{X}}_{\mathrm{CMI}_{10}}$ shares more than 99% information with the fill level $\mathbf{y}$. This causes the question: how much information is required to learn an accurate network? Hence, the learning curve when utilizing another amount of information and consequently a different number of proprioceptors is evaluated. Figure 6.12 illustrates the mean CEE (blue curve) of the softmax layer and the corresponding standard deviation (gray area) for five, ten and fifteen proprioceptive inputs. Each of these three different LSTM networks were trained twenty times to achieve meaningful results. As already shown (see Figure 6.8), five proprioceptors contain 63% information while fifteen proprioceptors share 100% information with the fill level. Since important information is missing, the learning curve when utilizing five proprioceptors converges to a mean CEE of about hundred. The resulting classifiers are not able to generate estimates with an adequate accuracy. In contrast to that, the estimation accuracy when utilizing ten or more proprioceptors converges to zero. Here, fifteen proprioceptors provide the fastest learning curve but also an increased risk of containing less relevant information. In particular, the five additional proprioceptors contain less than 1% of so far unknown information while increasing redundancy or add spurious correlations. Consequently, ten proprioceptors result in an appropriate trade-off between the input dimensionality and the learning curve.

Finally, the number of blocks contained in the LSTM network is evaluated. Utilizing more blocks allows remembering more information at once but also increases the number of connections. In turn, the effort for learning the adaptable weights of these connections is rising what strongly influences the computational demand. Here, a total of 20 LSTM networks with varying block configurations were learned for thousand epochs from the acquired training data $\tilde{\mathbf{X}}_{\mathrm{CMI}_{10}}$. Each of these configurations is trained ten times from randomly chosen initial weight configurations resulting in an overall of 200 learning procedures. The blue bars in Figure 6.13 represent the mean CEE over all epochs while the green bars represent the mean of the lowest CEE found. As can

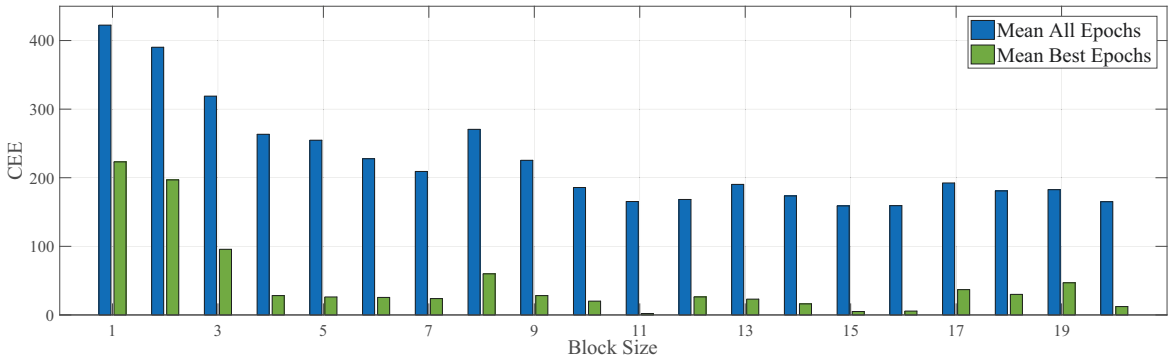


Fig. 6.13: The number of blocks is adapted in order to evaluate its influence on the overall learning curve. For this, the particular network's weight configuration was randomly initialized and trained for exactly thousand epochs. This was repeated ten times for each block configuration resulting in an overall of 200 learning procedures. The mean CEE over all epochs and repetitions (blue bars) and the mean of the lowest CEE for all repetitions (green bars) are shown. Less than four blocks are not able to store all relevant information. Utilizing seven blocks results in a similar learning curve as for ten blocks and further reduced the computational demand. The lowest CEEs are achieved by utilizing eleven blocks.

be seen, the block size strongly influences the learning curves of the corresponding networks which is explained as follows:

- Block size < 4: The memory of the network is too small to store all import information what therefore limits the accuracy of the results. This is similar to the usage of five proprioceptors which do not provide enough information (see Figure 6.12 left).
- Block size $4-7$: The learning curves are systematically increasing, whereby seven blocks already achieve results comparable to the usage of ten blocks contained in LSTM_{CMI}.
- Block size $8-11$: With eight blocks the network has a high probability to store irrelevant information which has a negative impact on the overall learning curve. This disturbing influence is slowly remedied by adding additional blocks. The highest accuracy of the overall evaluation is found for the networks constructed of eleven blocks.
- Block size > 11: The results are further oscillating and the accuracies are getting worse. This is due to storing irrelevant information but also to the increased network size which disables the learning algorithm to finish training within thousand epochs.

To summarize, the lowest CEE is achieved by the networks which utilize eleven blocks while seven blocks already produce good results. Latter has the advantage that the network is constructed of fewer connections which reduces the learning effort. Hence, the user has to define the block size with regard to the required accuracy and the available computing power.

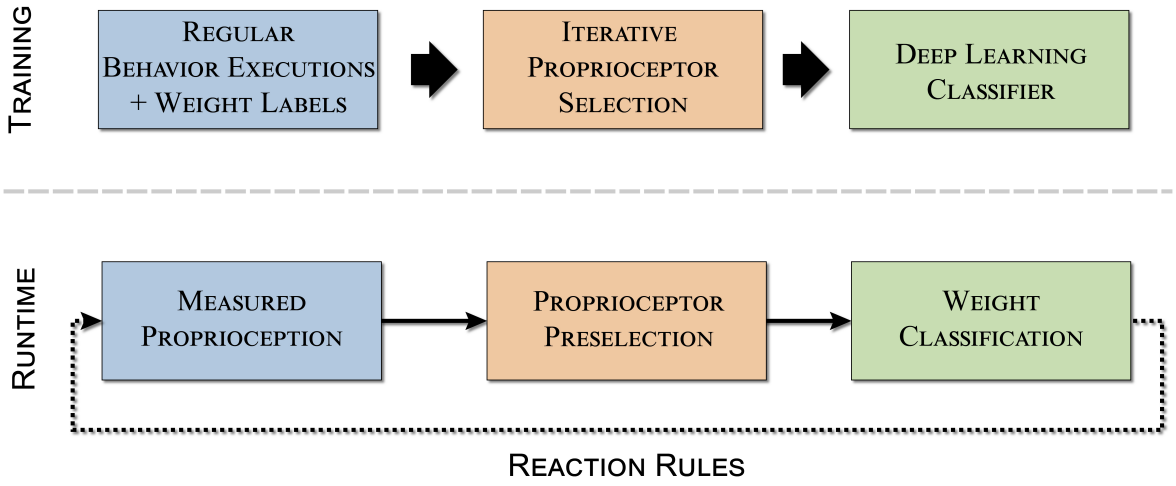


Fig. 6.14: Overview of the presented neural network based V-BSPM approach with regard to the methodology introduced in Section 4.3. Regular training data of the proprioception together with context specific labels of a water extraction station's fill level are recorded. To erase less relevant and redundant information, an optimal set of proprioceptors is selected iteratively. To this end, the mutual information between the proprioceptors and the weight is measured by means of CMI. This increases the deep learning efficiency of an LSTM network. Similar to a human guessing motion, the augmented proprioception gets more certain about the actual weight or fill level the longer the behavior is executed. As a result, the extraction process can be adapted by specific reaction rules.

6.2.6 Conclusion

This section applied the V-BSPM approach to classify the fill level of a self-contained water extraction station where an overview can be seen in Figure 6.14. Here, state-of-the-art FT sensors are not applicable due to various reasons, e.g., measurement accuracy and load capacity. These restrictions are intensified by the rough environmental conditions in the application area of underground mines. Instead, a neural network based V-BSPM is learned to augment the robot's measured proprioception with manually labeled context descriptions. In particular, a mobile robot platform equipped with a UR5 robotic arm places and lifts the water extraction station at arbitrary locations. Here, the regular proprioception of a five second lifting behavior at various positions and different fill levels is utilized for learning. For this, each behavior execution is assigned with a symbolic label which corresponds to the station's fill level (empty, half-full, full). At runtime, the V-BSPM augments the robot's proprioception with the most probable label where a certain decision can be met after half a second. This enables the robot to successfully complete the extraction procedure at arbitrary positions.

One advantage of such a classification task is that a final decision needs to be taken only after finishing the behavior. Hence, the deep learning classifier has the possibility to utilize all measurements gathered during behavior execution to increase its certainty. In particular, past measurements are taken into account by the indefinite memory of LSTM blocks. This allows the LSTM network to increase its accuracy the longer the behavior is executed which is a major advantage over classical RNNs. As a result, the V-BSPM can meet a certain decision about the fill level of the station after less than

half a second. To this end, the learning efficiency is increased by erasing less relevant and redundant information from the training data. Here, CMI in combination with an iterative proprioceptor selection procedure was applied. The remaining proprioceptors contain the maximum of mutual information with the target, i.e., the label of the fill level. This reduces the computational effort and allows learning of an accurate and generalizing V-BSPM from a small set of behavior examples.

A major advantage of the model-based V-BSPM is given by its constantly low reaction time which is also independent from the number of training examples. In particular, only the amount of network connections is influencing the computational demand required for input to output mapping. Hence, utilizing an adequate network configuration ensures the real-time augmentation of the robot's proprioception with a virtual force sensor. However, the conducted experiments highlight the immense effort spent for finding an accurate and generalizing LSTM configuration. Without any knowledge about such state-of-the-art deep learning architectures the user risks to generate an underfitted or overfitted network. This can be easily avoided by increasing the number of training and validation examples but requires an extended data acquisition and training phase. Hence, an adequate ratio of user experience and training effort is vital for efficiently learning the proposed V-BSPM.

6.3 Summary

This chapter presented two V-BSPM applications which augmented a robot's proprioception with virtual force sensors. For this, training examples of the measured proprioception are manually extended with context specific labels. The runtime proprioception is then augmented with an estimate of the actual label. This provides a UR5 robotic arm with force estimation capabilities which are comparable and even beyond state-of-the-art FT sensors. The robot can then physically interact with its environment by utilizing specified reaction rules. To this end, two applications were presented:

1. A lazy learning V-BSPM was applied for measuring the tightening torque exerted by the use of a custom torque wrench.

2. A deep learning V-BSPM was utilized for weight classification which corresponds to the fill level of a self-contained water extraction station.

Here, both applications benefit from the usage of information-theoretic measures which are applied to select a subset of proprioceptors for learning. More precisely, (1) utilizes information transfer while (2) is based on the mutual information between the particular target and the proprioceptors.

Furthermore, the lazy implementation of (1) utilizes an adapted SDTW algorithm which detects the most similar training data for a measured sequence of runtime data. In contrast, (2) utilizes the training data to learn a neural network which maps the robot's measured proprioception to the context labels. This network further benefits

from the indefinite memory of LSTM blocks. In particular, the sequence length of the SDTW algorithm is configured manually while LSTM blocks have the ability to memorize previous proprioceptive states indefinitely. A further drawback of (1) is that the computational complexity for the real-time augmentation depends on the size of training data. This limits the robot's interaction with the torque wrench to a predefined TCP position. In contrast, the real-time augmentation of (2) is independent from the amount of training data. Hence, regular behavior executions can be spread over the robot's workspace. The increased number of training examples enables the V-BSPM to generalize context labels for arbitrary TCP positions while still providing a constantly low reaction time.

So far, the recorded behavior examples were manually labeled by the user. This requires a fair degree of knowledge about the particular task, e.g., accurately configuring a torque wrench at a consistent grasping position. Here, a FT sensor can be utilized to simplify the acquisition phase which, due to the low-cost motivation, needs to be disassembled after training for regular operation. A further drawback of the presented V-BSPMs is that they are trained for regular behavior executions only. Hence, the user has to ensure that the runtime execution of the behavior is not influenced by extrinsic perturbations. This may be applicable when interaction takes place a static environment but is a major risk for safe physical interaction with humans. To address this challenge, learning a combination of multiple BSPMs for regular and perturbed behavior executions is introduced in the following.

7 Force/Torque-Estimation and Perturbation Detection in Physical Human-Robot Interaction

The ability to sense the environment is a vital requirement for intelligent and safe robotics. For instance, FT sensors can be used to measure external influences on a robot and, in turn, generate adequate responses. As mentioned before, such sensors are often expensive, require additional power supply and have a negative impact on the payload. Furthermore, they are not able to separate behavior-specific intrinsics from extrinsic perturbations.

The setup, in which the six-dimensional FT sensor FT150 is mounted within between the UR5 robotic arm and the three-fingered gripper, is shown in Figure 7.1. Here, the presented application focuses on safety in physical human-robot interaction. In particular, the robot picks and places three different shapes which need to be provided in the correct order by a human coworker. The FT values are utilized to trigger the handover of the objects and to detect collisions. The latter mostly occur due to a wrong order of the shapes but can also be triggered by a haptic misconduct of the human coworker.

The BSPM approach is applied to learn a FT sensor which provides similar and even more accurate sensing capabilities without the drawbacks of additional hardware. For this, two particularly different BSPMs are combined. An F-BSPM which predicts the behavior-specific intrinsics and a V-BSPM which approximates the total measurements of a FT sensors. First allows to detect extrinsic perturbations while second augments the robot's proprioception with a virtual FT sensor. For this, training examples from regular and perturbed behavior executions are recorded. Next, the dimensionality of the proprioceptors is reduced by means of information theory. The BSPMs are then implemented by utilizing ANNs which among other advantages also provide a constantly low reaction time. This is vital for the safety of the presented application. Here, the applicability of the presented approach is demonstrated by utilizing only well-known ANN architectures with very limited complexity. In contrast to deep learning techniques (see Section 6.2), these networks require less parametrization effort which makes it easier to use by non-experts. At runtime the FT sensor is unplugged and instead the F-BSPM and V-BSPM are utilized to implement a collision detection approach which allows an intuitive and safe physical interaction with the human coworker. In the following sections, the proposed combination and the particular details of both BSPMs are explained in more detail.

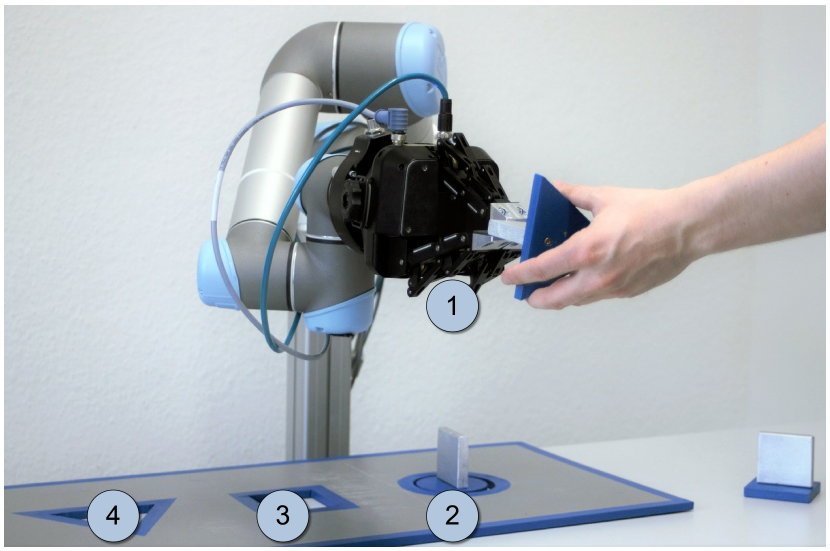


Fig. 7.1: A human coworker is handing over different shapes to the robot which therefore picks and places them in a predefined order $(1, 2, 1, 3, 1, 4, 1)$. For safe behavior execution, the robot must be able to reliably detect perturbations resulting from unexpected physical interactions with the environment and the human coworker. For training, a three-dimensional FT sensor is mounted within between the UR5 robotic arm and the three-fingered gripper. The basic idea is to learn a combination of multiple BSPMs from this FT sensor. At runtime, the functionality of the FT sensor is replaced by these models. Besides accurate FT measurements, this further enables the robot to accurately detect unexpected collisions.

7.1 Data Acquisition: Regular and Perturbed Behavior Examples

The behavior is designed with regard to the task shown in Figure 7.1. Here, the human coworker is handing over different shapes which are picked and placed by the robot. In order to provide training data, the behavior is conducted without performing the interaction. Instead, the robot moves from the handover position (1) to the first placing position (2) and back to position (1) after a short offset. This is also done for position (3) and (4) and results in a behavior with the following motion trajectory $(1, 2, 1, 3, 1, 4, 1)$.

This trajectory is repeatedly executed for 60 seconds resulting in a total of $n = 7500$ samples $\mathbf{x} = (x_1, \ldots, x_m)$ which contain about two executions of the trajectory. Here, each sample contains the $m = 105$ raw proprioceptive readings of the UR5 robot. For the presented task, the learning target is derived from the simultaneously recorded six-dimensional output of the FT sensor $\mathbf{y} = (y_1, \ldots, y_6)$ where $\{y_1, y_2, y_3\}$ are force measurements and $\{y_4, y_5, y_6\}$ contain torque values. The FT150 data rate is about 100 Hz which is less than the UR5 robot's 125 Hz. Hence, intermediate values are generated by utilizing the introduced DMD interpolation procedure (see Section 5.2.2 p. 75). Furthermore, the continuously increasing relative time w is recorded for the

later correlation analysis. The resulting training data has the following appearance.

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}, \mathbf{Y} = \begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_n \end{pmatrix}, \mathbf{w}_1 = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}. \tag{7.1}$$

In contrast to the previous applications, this data acquisition process is repeated three times under very specific circumstances.

1. Regular training data $\{\mathbf{X}_R, \mathbf{Y}_R, \mathbf{w}_R\}$: No extrinsic perturbations are influencing the behavior execution of the robot and therefore the recording contains only intrinsics.

2. Perturbed training data $\{\mathbf{X}_P, \mathbf{Y}_P, \mathbf{w}_P\}$: the robot is almost permanently perturbed by manually applied forces during behavior execution to result in a combination of intrinsics and extrinsics.

3. Validation data $\{\mathbf{X}_V, \mathbf{Y}_V, \mathbf{w}_V\}$: the first half of the recording is executed under regular conditions while the human continuously applies forces during the second half.

These different sets of training data are required to learn and evaluate the learning process. In particular, only regular training data is utilized to learn the F-BSPM while a combination of regular and perturbed training data is required for the V-BSPM. The latter enables the V-BSPM to approximate the total proprioception which combines intrinsic and extrinsic influences. This is a major advantage to previous V-BSPM applications (see Chapter 6) which was learned from regular training examples only. To increase the efficiency of the corresponding learning process, the recorded proprioceptive measurements are normalized by utilizing the introduced z-score (see Section 2.2).

7.2 Proprioceptor Selection

As stated above, the interface of the UR5 provides $m = 105$ proprioceptive measurements, containing a variety of different information sources such as set, control and actually measured values of the robot's joints. Some of these sources are relevant for learning the corresponding BSPMs while others are non-informative. Here, proprioceptors which are beneficial for predicting the FT values are detected by utilizing Transfer Entropy (TE). To achieve meaningful TE results, the behavior examples are converted to the same level of magnitude by applying a four bit binning estimator.

As stated above, two different BSPMs are generated which also involve different sets of training data. In particular, the F-BSPM is learned from regular examples $\{\mathbf{X}_R, \mathbf{Y}_R, \mathbf{w}_R\}$ where the proprioceptive measurements $\mathbf{X}_R$ and the FT values $\mathbf{Y}_R$ represent the intrinsics of the behavior. These intrinsics depend on the actual phase of the behavior execution which is represented by the relative execution time $\mathbf{w}_R$. Here,

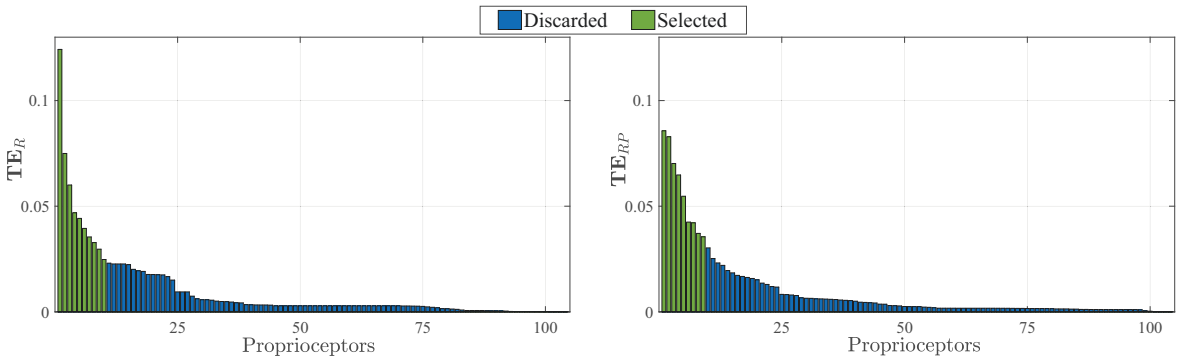


Fig. 7.2: The normalized and ordered TE values are utilized to select the most beneficial proprioceptors. Left: The information transfer $\mathbf{TE}_R$ from the proprioceptors to the relative time is utilized to select ten proprioceptors for learning the F-BSPM. Right: The information transfer $\mathbf{TE}_{RP}$ from the proprioceptors to the FT values identifies nine proprioceptors which are assumed to be beneficial for learning the V-BSPM.

TE is used to measure the information transfer between the proprioceptors and the relative time by

$$\mathbf{TE}_R = T_{\mathbf{X}_R \to \mathbf{w}_R}. \tag{7.2}$$

In contrast to that, the V-BSPM is learned from a combination of the regular and perturbed training data. Here, the UR5 robot's proprioception $\{\mathbf{X}_R, \mathbf{X}_P\}$ and the FT150 FT values $\{\mathbf{Y}_R, \mathbf{Y}_P\}$ contain influences from intrinsics and extrinsic perturbations. Their combination therefore does not exclusively depend on the phase of the behavior execution. Consequently, instead of applying TE with respect to the relative time, it is important to extract proprioceptors which contain information about intrinsics and extrinsics. In particular, TE is applied to detect correlations between the proprioceptors and the FT values by

$$\mathbf{TE}_{RP} = T_{\{\mathbf{X}_R, \mathbf{X}_P\} \to \{\mathbf{Y}_R, \mathbf{Y}_P\}}, \tag{7.3}$$

where the six-dimensional FT measurements $\{y_1, \ldots, y_6\}$ are represented by their joint intersection.

Figure 7.2 shows the normalized and ordered TE values of $\mathbf{TE}_R$ (left) and $\mathbf{TE}_{RP}$ (right). Here, ten proprioceptors which combine more than 50% of the overall TE contained in $\mathbf{TE}_R$ are selected for learning the F-BSPM. The remaining training data $\tilde{\mathbf{X}}_R$ contain the control values of the robot's joint states (e.g. position and velocity), which are independent from extrinsics and are good predictors for the actual phase of the behavior. In contrast, $\mathbf{TE}_{RP}$ identifies nine proprioceptors which account for more than 50% of the overall TE. Here, the combination of the regular $\tilde{\mathbf{X}}_R$ and perturbed $\tilde{\mathbf{X}}_{RP}$ training data $\tilde{\mathbf{X}}_{RP}$ includes 120 seconds of measured values such as the robot's motor currents. These proprioceptors are affected by intrinsics and extrinsic perturbations which enables the V-BSPM to estimate FT measurements for both regular and perturbed behavior executions.

TE is utilized to identify information about future changes of the time $\mathbf{w}_R$ and regular/perturbed FT values $\{\mathbf{Y}_R, \mathbf{Y}_P\}$. Here, the selected proprioceptors $\tilde{\mathbf{X}}_R$ increase the predictability of intrinsic FT values while $\mathbf{TE}_{RP}$ is beneficial for estimating the total

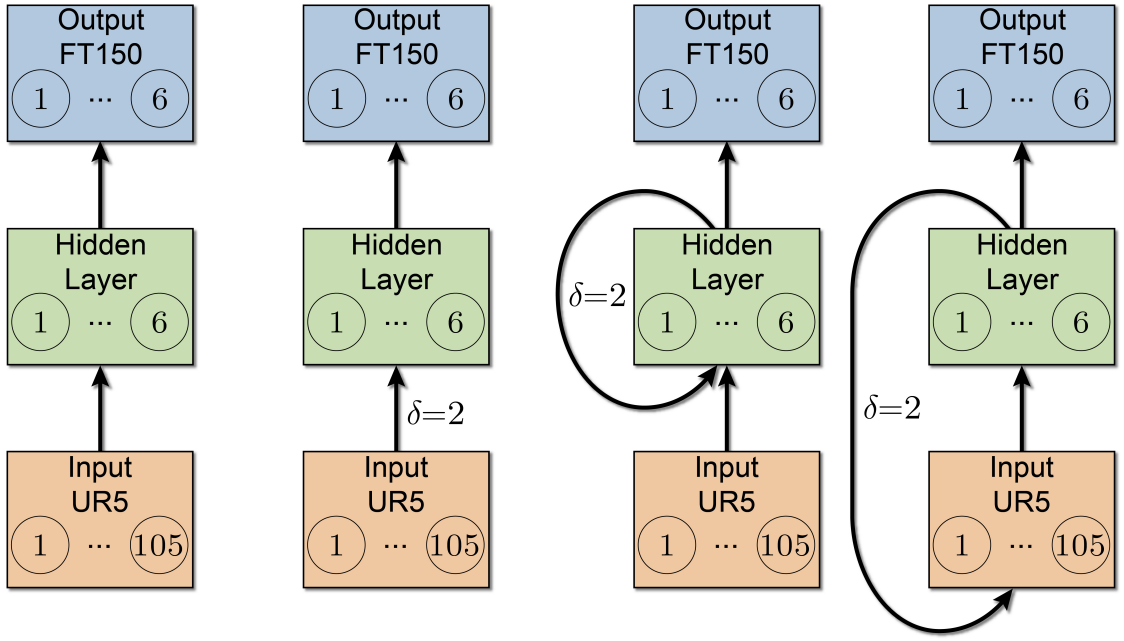


Fig. 7.3: Utilizing the same basic configuration, different network architectures are employed for learning the BSPMs. FF does not contain recurrent connections and therefore has no short-term memory while TD remembers information by utilizing a constant amount of previous inputs. In contrast to that, REC supports layer internal recurrence while NARX extend the advantages of TD by utilizing previous predictions to increase the accuracy of the actual one.

FT values. In contrast to the previous application, no iterative selection procedure is applied where the selected proprioceptors can contain redundant parts of information. For the proposed task, redundant parts of information help to filter spontaneous noise during continuously estimating FT values. This increases the accuracy of the utilized ANN architectures.

7.3 Learning Neural Networks

The FT values strongly differ during behavior execution. These fluctuations occur due to the behavior execution but are also triggered by spontaneous influences of extrinsic perturbations. Hence, an accurate short-term memory is required while an indefinite memory is less beneficial. This has the advantage that the complexity of the network architecture can be reduced. In particular, deep learning approaches such as LSTM blocks are not required for learning the BSPMs. This reduces the configuration effort and therefore increases the applicability of the overall approach.

As illustrated in Figure 7.3, several well-known ANN architectures are constructed to map the 105-dimensional proprioceptive input data to the six-dimensional FT values. According to Heaton [130], one hidden layer is sufficient to solve such a continuous mapping between two finite spaces. Hence, each network contains one hidden layer and is further generated with the same basic settings:

- One hidden layer which is fully connected with the input and output layer.
- Six hidden neurons with sigmoid activation functions.
- Six output neurons with linear activation functions.

This ensures the comparability of the results while the straightforward configuration increases the reproducibility by non-experts.

To this end, a *feed-forward* (FF), *time-delay* (TD), *recurrent* (REC) and *nonlinear autoregressive network with exogenous inputs* (NARX) are employed. FF and REC are similar to the definition of FNNs and RNNs introduced in Section 3.2.2 while TD [131] and NARX [132] are shortly examined here. A TD supports no recurrent connections and therefore is most is similar to classical feed-forward neural networks. In contrast to FF, it has additional connections to a constant amount of previous inputs where the activation value of neurons is calculated by the weighted sum of previous activations. Consequently, neurons have the ability to react with a certain time delay what gives the entire network a finite memory. NARX networks require recurrent connections by definition and therefore are a special kind of RNN. In more detail, together with an independent (exogenous) input, a NARX utilize a finite amount of its own outputs (autoregressive) to predict the actual output. This kind of recurrence allows the network to take into account short-term dependencies which prevents the prediction from sudden changes and results in smooth outputs. Independent from the particular architecture, expect for FF, the time delay δ defines a network's memory. Here, the delay is set to $\delta = 2$ by default what enables the particular networks to remember the last two inputs (TD), internal states (REC) or outputs (NARX).

The basic idea is that this short-term memory benefits from the application of the applied proprioceptor selection. In particular, the recurrent connections allow the network to take into account several past states. The uncertainty about its future state is then reduced by the proprioceptors which transfer the highest amount of information to its future. Hence, the network architecture is motivated by the prediction task which needs to resolve the uncertainty about the future. This is related to the definition of TE. Furthermore, redundant proprioceptive information allows filtering spontaneous noise what is less relevant for an indefinite memory. In the following, the benefits of proprioceptor selection are evaluated for learning the F-BSPM and V-BSPM.

7.3.1 F-BSPM Learning

The F-BSPM has the goal to predict the intrinsics of the FT sensor also in the presence of extrinsic perturbations. Hence, the prediction needs to be independent from the amount of extrinsic perturbations. For now, all proprioceptors $\mathbf{X}_R$ are utilized for learning the F-BSPM by applying the gradient based backpropagation of error technique on regular training data. In particular, a maximum of 100 epochs is conducted by means of offline learning.

The accuracy of the trained networks is then evaluated by utilizing the semi-perturbed validation data $\mathbf{X}_V$. Utilizing all proprioceptors results in the F-BSPM

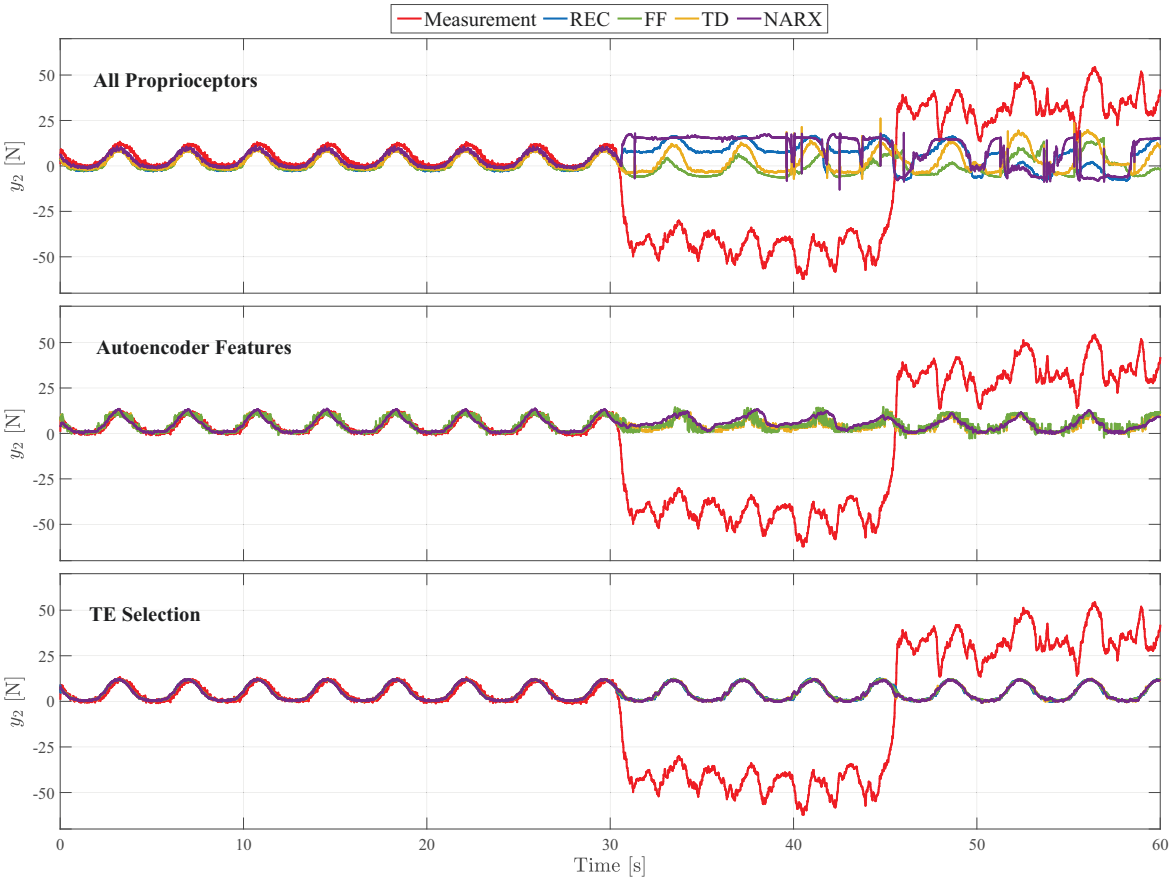


Fig. 7.4: The F-BSPM predictions of $y_2 \in \mathbf{Y}_V$ for network architectures which was trained with $\mathbf{X}_R$. Using all proprioceptors (top), autoencoder features (middle), or the proposed TE based proprioceptor selection (bottom) influences the accuracy of the predictions. Here, only the TE selection is able to suppress the extrinsic perturbations which occur in the second half of the recording $\mathbf{X}_V$.

predictions for $y_2 \in \mathbf{Y}_V$ shown in Figure 7.4 top. As long as no perturbation occurs, all networks are approximately predicting the correct value. However, during the perturbed half of the behavior execution, all networks fail to predict the correct intrinsics. This is due to the fact that the 105 input proprioceptors contain non-relevant information that obfuscate important relations. Instead, the subset of proprioceptors which was obtained from the proposed selection procedure is compared to the usage of autoencoders [133].

Autoencoders are a special kind of ANNs which are gaining popularity in the field of dimensionality reduction and feature extraction. Such an autoencoder consists of an encoder and a decoder component, where both of which are neural networks. More precisely, the encoder maps the input data to a smaller set of hidden neurons while the decoder tries to reconstruct the original input. This process can be stacked to reduce the dimensionality of the input data through a stepwise layering. Here, the regular training data $\mathbf{X}_R$ is reduced from 105 input dimensions to 50 dimensions by the first encoder while stacking a second one results in a subset of ten dimensions. Similar to the proposed ANN architectures, the encoding process utilizes a sigmoid function while a linear function was implemented in the decoders. As a result, the 105-

Tab. 7.1: The MSEs of the V-BSPM resulting from different inputs and neural network architectures. Here, the REC network trained with the submatrix $\tilde{\mathbf{X}}_{RP}$ achieves the most accurate predictions of $\mathbf{Y}_V$ for a validation with the semi-perturbed recording $\tilde{\mathbf{X}}_V$.

	All Proprioceptors	**Autoencoder**	**TE Selection**
FF	155.24	72.62	25.14
TD	95.48	81.45	14.82
NARX	22.43	62.34	7.33
REC	13.74	42.48	**3.41**

dimensional training data is reduced to a ten-dimensional feature data set. This feature data is then used to train the different neural networks. The resulting predictions for $y_2 \in \mathbf{Y}_V$ can be seen in Figure 7.4 middle. Especially during the perturbation phase, the accuracy increases compared to all proprioceptors. Another advantage is the decrease of computational effort spent for learning which is induced by the reduced amount of inputs. Similar to the usage of the PCA, a drawback of autoencoders is that temporal influences and correlations to the learning target are not taken into account. As a result, potentially useful information is discarded.

Finally, the submatrix $\tilde{\mathbf{X}}_R$ which was selected by means of TE is utilized for learning the F-BSPM. In contrast to the previous results, as shown in Figure 7.4 bottom, most of the proposed networks are able to accurately predict the intrinsics for $y_2 \in \mathbf{Y}_V$ from the ten remaining proprioceptors. Also during extrinsic perturbations the estimated intrinsics is not influenced. Here, the highest accuracy is achieved by utilizing the REC architecture which results in a MSE of 0.05 N for the first half of the recording. In the following, this network is referred to as F-BSPM$_{\text{REC}}$.

At runtime, the F-BSPM$_{\text{REC}}$ predicts the FT sensor intrinsics from the selected proprioceptors of the UR5 robot. These predictions can be compared to the actually measured FT values what allows to distinguish intrinsics from extrinsics and further gives an estimate of the amount and direction of extrinsic perturbations. To further avoid the usage of the FT150 during runtime, the V-BSPM is learned in the following.

7.3.2 V-BSPM Learning

The V-BSPM aims at predicting absolute FT values of the FT150 whether the measurements are triggered by behavior-intrinsics or extrinsic perturbations. For this, the combination of regular and perturbed data $\{\mathbf{X}_R, \mathbf{X}_P\}$, which combines intrinsics and extrinsics, is used for training with a maximum of 100 offline learning epochs. The estimation accuracy of the different ANN networks is then evaluated by utilizing the semi-perturbed recording $\mathbf{X}_V$. The resulting MSE for the different ANN predictions of $y_2 \in \mathbf{Y}_V$ are shown in Table 7.1.

As can be seen, the best results are achieved when utilizing the proprioceptors selected with TE. This is due to the fact that TE measures additional information which is not already contained in the FT measurements of the training data (Section 3.1.3).

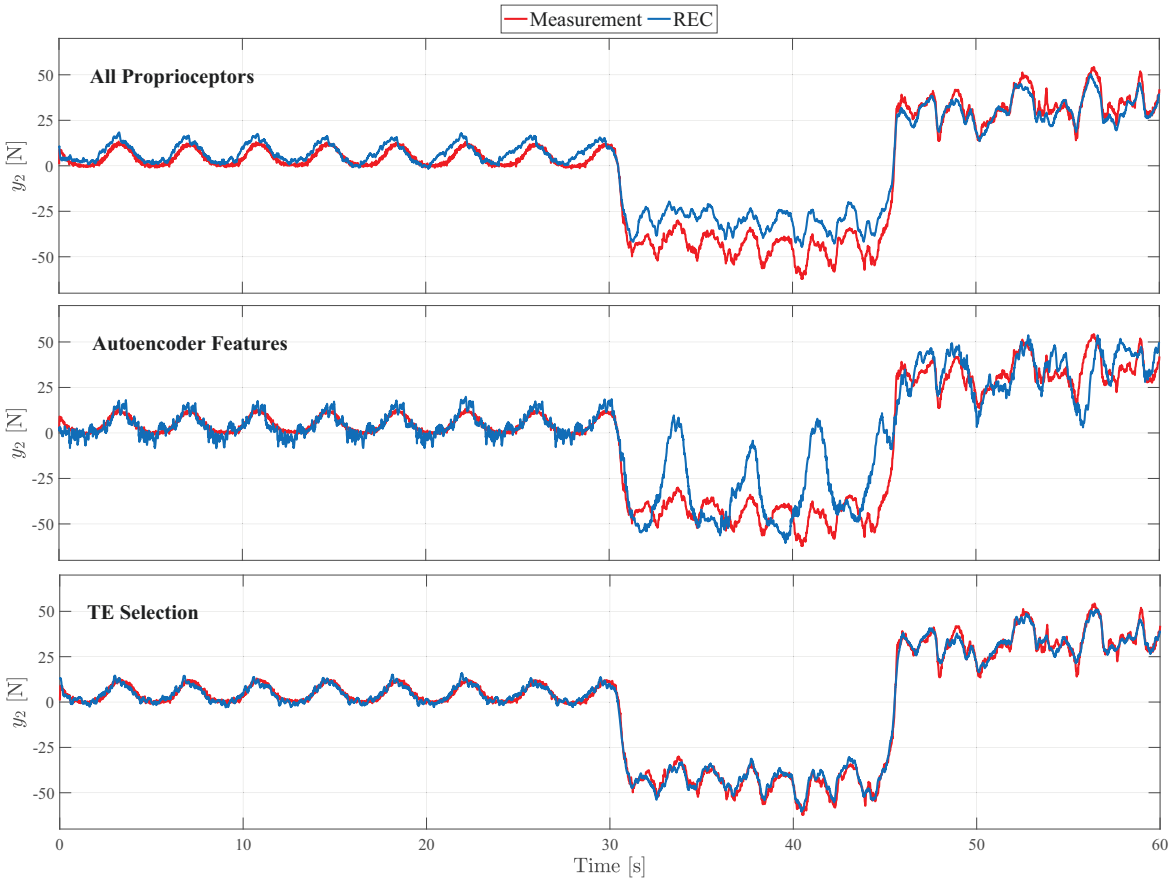


Fig. 7.5: The predictions of $y_2 \in \mathbf{Y}_V$ for the recurrent network REC trained with the TE selection of the regular and perturbed training data $\tilde{\mathbf{X}}_{RP}$. Using all proprioceptors (top), autoencoder features (middle), or the proposed TE based selection (bottom) influences the accuracy of the V-BSPM. Only the proprioceptors selected by utilizing **TEf** are able to correctly estimate the total force during regular and perturbed execution of the behavior.

Hence, the selected proprioceptors are good predictors when the actual FT measurements are known. Since the FT sensor is not available during, this can be achieved by utilizing recurrent connections in the neural network. For example, NARX predicts FT measurements by combining δ previous predictions with the actually measured proprioception. As a result, REC and NARX utilize previous state information about virtual FT measurements and perform better than FF and TD.

Furthermore, the accuracies of REC and NARX are decreased when using autoencoder features as input. A possible explanation for this effect is that the objective function of autoencoders only focuses on the amount of information retained by using the generated features. This is detrimental in various physical tasks in which some proprioceptors have limited variability but strong influence on the task.

However, the best result is obtained by using the training data obtained by the TE based selection procedure $\tilde{\mathbf{X}}_{RP}$ and the REC network architecture. Predictions of $y_2 \in \mathbf{Y}_V$ generated by REC are compared to using all proprioceptors and autoencoder features in Figure 7.5. Given these results, REC is utilized to determine the output of the V-BSPM and is referred to as V-BSPM$_{\text{REC}}$.

At runtime, V-BSPM$_{\text{REC}}$ augments the robot's proprioception with estimates of the

total FT measurements. In particular, actual FT values are approximated from past proprioceptive measurements of the UR5 robot. This enables the approach to be independent from additional hardware such as the FT150.

7.4 Virtual Force Sensor and Perturbation Detection

The FT150 is unplugged and replaced by a virtual counterpart during runtime. For this, the subsets of proprioceptors are selected from the real-time interface of the UR5 robot and fed into the corresponding BSPMs. In particular, the virtualized sensor is constructed by the V-BSPM$_{\text{REC}}$ and F-BSPM$_{\text{REC}}$. The V-BSPM$_{\text{REC}}$ augments the robot's proprioception with approximated FT values while the F-BSPM$_{\text{REC}}$ predicts the phase dependent intrinsics. The amount and direction of extrinsic perturbation can then be determined by

$$\text{BSPM}_{\text{EXT}} = \text{VBSPM}_{\text{REC}} - \text{FBSPM}_{\text{REC}}. \tag{7.4}$$

As a result, the proposed method can be applied during runtime without making use of the FT sensor in order to estimate total, intrinsic and in consequence external FT values from previous experience. This reduces costs, power consumption, etc. but is no mandatory prerequisite. For example, an additional failure protection can be implemented by comparing the FT150 measurements to the V-BSPM$_{\text{REC}}$.

The resulting estimates represent extrinsic perturbations which are applied by humans, collisions or other external factors and need to be treated with specified reaction rules. For instance, safely detecting these collisions enables the human coworker to intuitively interact with the robot or can be utilized to implement an emergency stop. A video of training and utilizing these BSPMs for the introduced shape placing task can be found under the following link: `https://youtu.be/60ue0X25S6k`.

To this end, the virtualized FT sensor is used to substantially increase the UR5 robot's interaction capabilities by the following reaction rules.

- $y_2 \in \text{BSPM}_{\text{EXT}} > 10\,\text{N}$: Pick the object and start the placing behavior when the human coworker slightly pushes the gripper while resting in the handover position.
- $|y_1, y_2, y_3| \in \text{BSPM}_{\text{EXT}} > 10\,\text{N}$: Return the object to the initial position whenever colliding with the environment during behavior execution.
- $y_2 \in \text{BSPM}_{\text{EXT}} < -10\,\text{N}$: Release the object when the human coworker slightly pulls the gripper while resting in the handover position.

Furthermore, the virtual FT sensor is utilized for the implementation of an emergency stop approach which outperforms the pre-built solution of the UR5. In the following, different experiments are conducted to evaluate the proposed combination of F-BSPM and V-BSPM.

7.5 Evaluation

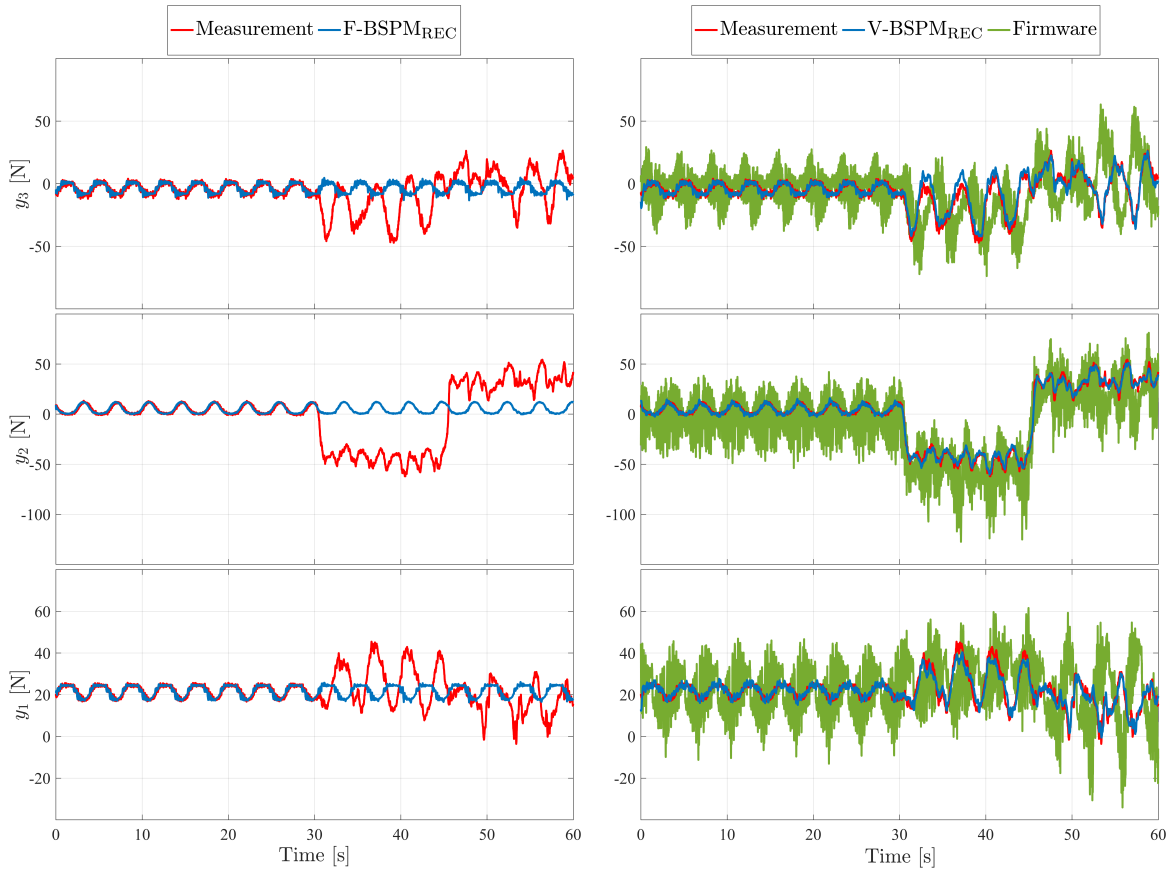


Fig. 7.6: The intrinsic predictions compared to the total approximations of the force measurements contained in the semi-perturbed validation data $\mathbf{X}_V$. Left: The intrinsic force predictions of F-BSPM$_{\text{REC}}$ are accurately approximating the unperturbed periods of the behavior execution and are further not influenced by extrinsic perturbations. Right: The total force approximations of V-BSPM$_{\text{REC}}$ provide a more accurate approximation of the FT150 force measurements than the firmware of the UR5 robot.

First, the accuracy of the virtualized FT sensor is evaluated and compared to the build-in FT estimator of the UR5 which is referred to as *firmware*. More precisely, the firmware makes use of joint torques and a kinematic model of the robot to predict the FT values at the TCP. In order to get comparable results, the firmware is calibrated to the mass and size of the FT150. Based on this calibration, the manufacturer specifies a TCP force accuracy of 25 N and a detection time of 250 ms for their pre-built solution.

First, V-BSPM$_{\text{REC}}$ is evaluated by investigating the MAE in comparison to the ground truth of the FT150 and the firmware. The resulting three-dimensional forces for the semi-perturbed validation data $\mathbf{X}_V$ are shown in Figure 7.6 right. The estimates provided by the firmware follow the general trend but exhibit significant noise. In contrast, the V-BSPM$_{\text{REC}}$ approximations are close to the measured ground truth of the FT150. The accuracy of the firmware sensor resulted in a MAE of 26.74 N which is slightly worse than the 25 N specified by the manufacturer. Even considering a time delay of 250 ms did not decrease the MAE score. In comparison, the MAE of the V-BSPM$_{\text{REC}}$

is 3.83 N with an average computation time of 5 ms (conducted on a dual core with 3.2 GHz). Also the torque estimates of the V-BSPM$_{\text{REC}}$ achieve a MAE of 1.07 N m what outperforms the firmware MAE of 5.67 N m.

Due to safety issues and to keep the robot reactive from the beginning, a minimal delay of $\delta = 2$ was utilized. To achieve a higher accuracy, the short-term memory of the network needs to be expanded. This results in an increased computational effort for training and runtime predictions. In particular, the short-term memory of the REC network is set to $\delta = 125$. This reduces the MAE of the force estimates to 1.94 N but also requires an adapted selection procedure. More precisely, the most relevant proprioceptors with a maximal delay of one second are selected by means of delayed TE (see Equation 3.16 p. 35)

$$\begin{aligned} \mathbf{TE}_R &= T_{\mathbf{X}_R \to \mathbf{w}_R}(125), \\ \mathbf{TE}_{RP} &= T_{\{\mathbf{X}_R, \mathbf{X}_P\} \to \{\mathbf{Y}_R, \mathbf{Y}_P\}}(125). \end{aligned} \tag{7.5}$$

As a result, the trained model requires an initial offset of one second to start the prediction while the detection time increases to an average of 10 ms.

In contrast to the proposed approach, the firmware of the UR5 and also the FT150 have no possibility to distinguish between intrinsics and extrinsics. The F-BSPM$_{\text{REC}}$ predicts the FT intrinsics in order to differentiate extrinsic perturbations from the V-BSPM$_{\text{REC}}$ FT approximations. Here, the F-BSPM$_{\text{REC}}$ is applied to the semi-perturbed data set $\mathbf{X}_V$. Figure 7.6 left shows the resulting intrinsic force predictions for $\{y_1, y_2, y_3\} \in \mathbf{Y}_V$. As can be seen, the intrinsics are accurately predicted during the unperturbed half of the recording and are also not affected by extrinsic perturbations in its second half.

Extrinsic perturbations BSPM$_{\text{EXT}}$ are then derived by the difference between VBSPM$_{\text{REC}}$ and FBSPM$_{\text{REC}}$. Figure 7.7 shows the amount and direction of extrinsic forces during the execution of the semi-perturbed validation data $\mathbf{X}_V$. The fluctuations in the first half of the execution are originated by the noise of both BSPMs which has an overall MSE of about 4 N. The particular accuracy can be used to define a threshold value. Here, a threshold of 10 N results in a precise distinction between model induced noise and extrinsic perturbations. This further allows neglecting the weight of the carried objects, which for the proposed shapes is about 0.6 N. Heavier objects can be taken into account by additional offsets or by extending the learning phase for objects of different weights.

Finally, the correct assignment and the particular reaction to these perturbations depend on task-specific reaction rules. For the proposed shape placing application several predefined reaction rules and an emergency stop approach are implemented and shown in the mentioned video.

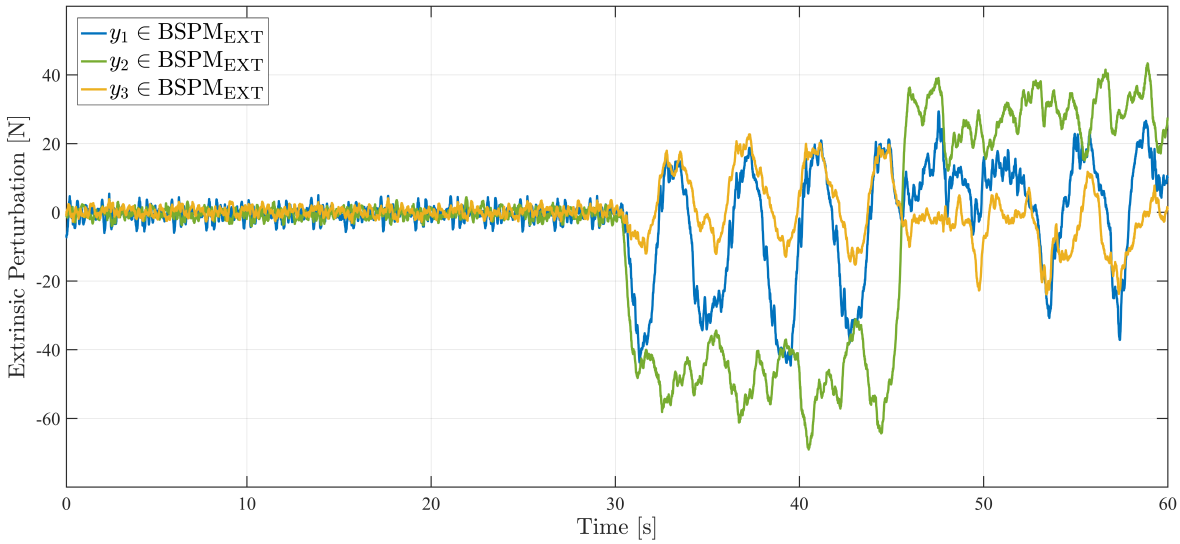


Fig. 7.7: The amount and direction of extrinsic forces during the execution of the semi-perturbed validation data $\mathbf{X}_V$. These three-dimensional forces are defined by the difference between V-BSPM$_{\text{REC}}$ and F-BSPM$_{\text{REC}}$. This enables the UR5 robot to interact with the human coworker by utilizing specified reaction rules.

7.6 Conclusion

A combination of F-BSPM and V-BSPM was applied to augment the robot's proprioception with a virtual FT sensor. This virtual FT Sensor outperforms the firmware of the UR5 robot which is based on an integrated mass acceleration model. More precisely, the resulting perturbation estimation approach is five times more accurate and significantly faster. Unlike the integrated firmware and the original hardware (FT150), their virtual counterpart further distinguishes intrinsic fluctuations from extrinsic perturbations.

One could argue that removing the hardware of the FT sensor results in an increase of the estimation error of the V-BSPM. That is certainly true but can be easily solved depending on the particular objective. For example, removing the FT150 changes the weight and kinematic chain of the robot and is detected as a constant extrinsic perturbation. This offset can be eliminated by a single initialization step or by adding a placeholder to the robot which has the same size and weight as the FT sensor.

An overview about this combined BSPM approach is given in Figure 7.8. Here, the behavior is executed under regular conditions while recording the measurements of the FT150 and the proprioceptors of the UR5 robotic arm. Hence, the recorded data contains only behavior-specific intrinsics. This data is used to learn the F-BSPM from the subset of proprioceptors containing the highest amount of information about the actual phase of the behavior execution. The intrinsic predictions of the F-BSPM are then deducted from runtime FT measurements to extract the amount and direction of extrinsic perturbations. To further remove the FT150 during runtime, the total sensor measurements are augmented by learning a V-BSPM. In contrast to intrinsic predictions, this V-BSPM is trained from data recorded during regular and perturbed behavior executions. The amount and direction of extrinsic perturbations is then de-

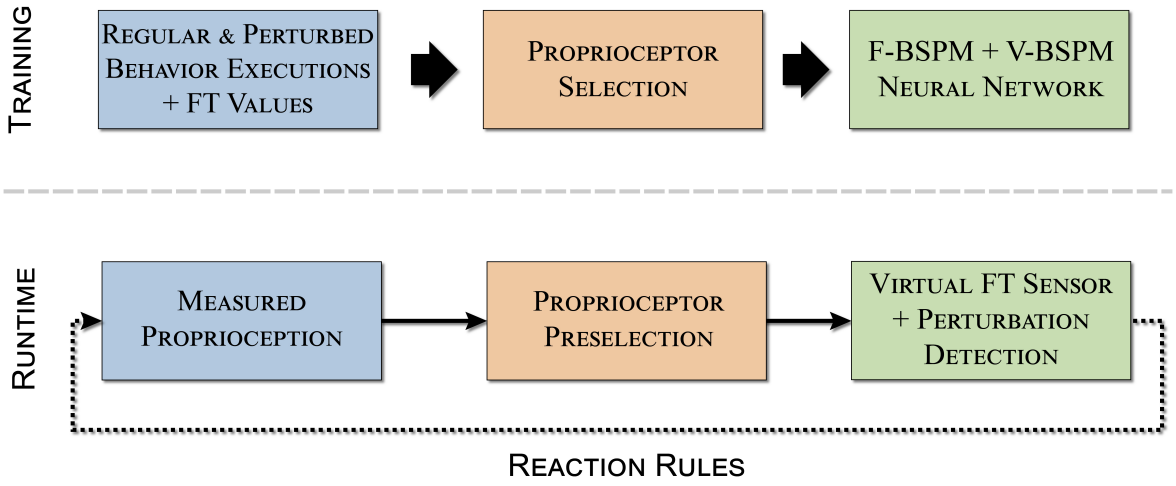


Fig. 7.8: Overview of the presented combination of F-BSPM and V-BSPM with regard to the methodology introduced in Section 4.3. Here, regular and perturbed behavior executions together with FT measurements are recorded as training data. Proprioceptors which are good predictors for the phase and FT values are then selected by means of TE. The remaining proprioceptors are used for neural network based learning of short-term memories. In particular, the F-BSPM predicts the intrinsics from the phase related proprioceptors while relations to FT values are beneficial for learning the V-BSPM. The latter allows estimating the total proprioception of the virtual FT sensor. At runtime, the difference between intrinsics and total FT values can be used to detect the amount and direction of extrinsic perturbations. This enables the robot to apply task-specific reaction rules and consequently increases its safety and interaction capabilities.

rived from the difference between the V-BSPM and F-BSPM predictions. Hence, a virtual replacement of the FT sensor is learned. The BSPMs further enable the approach to distinguish between intrinsics and extrinsics what is an advantage over the original FT sensor and the integrated firmware of the UR5. For the proposed shape placing task, the FT150 is unplugged while the virtual FT sensor is utilized to safely detect collisions. As a result, the robot is augmented with force sensing capabilities which allow safe and intuitive human-robot interaction.

To this end, classical recurrent network architectures provide the best results. This is attributed to the TE based proprioceptor selection procedure which involves a related information processing. In contrast to the previous application of deep learning, the approach was exclusively implemented by well-known short-term memory architectures. Here, limiting the complexity of these networks decreases the computational effort required for the corresponding learning procedure and ensures the comparability and reproducibility of the results. This further decreases the effort and knowledge required for defining an adequate network configuration. I.e., applying more complex architectures, larger time delays or even state-of-the-art deep learning architectures are promising to improve the proprioception capabilities but would decrease the applicability to non-experts.

8 Conclusion

This chapter summarizes the objectives and results of the presented thesis. First, the main contributions made to the field of proprioception models for robotic force estimation are recapitulated. Then, future directions and possible extensions are discussed before concluding with several final remarks.

8.1 Main Contributions

Robots that support humans in physically demanding tasks require accurate force sensing capabilities. A common way to achieve this is by monitoring the interaction with the environment directly with dedicated force sensors. Major drawbacks of such special purpose sensors are the increased costs and the reduced payload of the robot platform. Instead, this thesis investigated how the functionality of such sensors can be approximated by utilizing force estimation approaches. Most of todays robots are equipped with rich proprioceptive sensing capabilities where even a robotic arm, e.g., the UR5, provides access to more than hundred sensor readings. Following this trend, it is getting feasible to utilize a wide variety of sensors for force estimation purposes. Human proprioception allows estimating forces such as the weight of an object by prior experience about sensory-motor patterns. Applying a similar approach to robots enables them to learn from previous demonstrations without the need of dedicated force sensors.

This thesis introduces *behavior-specific proprioception models*, a novel concept for enhancing robotic behavior with estimates of the expected proprioceptive feedback. These models extend classical programming by demonstrations methods where only movement data is learned and enable robots to estimate forces during behavior execution. In the neurobiological literature proprioceptive forward and inverse models are considered. The introduced concepts of F-BSPMs and I-BSPMs can be understood as an operationalization of these proprioceptive models for the application in robotics. F-BSPMs predict the expected proprioception which can be compared to the measured proprioception. This allows extracting extrinsic perturbations in the space of the particular proprioceptor. For example, during walking a robot predicts its stability which continuously changes due to the execution of walking itself. Hence, guiding forces of a human interactant can be derived from the difference between the expected and actually measured stability. In contrast, I-BSPMs are used to directly estimate behavior parameter such as the step length of the walking behavior. This simplifies the formulation of the corresponding reaction rules where the exact amount and direction of forces is not relevant for the compensation of extrinsic perturbations.

In addition, this thesis introduced V-BSPMs which augment the robot's proprioception with virtual sensors. This resembles the capabilities of skilled mechanics who can estimate forces from previously experienced effects on their proprioception. For example, virtual sensors are able to estimate force values from their effects on the robot's joint currents. Furthermore, multiple BSPMs can be combined to enhance the functionality of special purpose sensors. This was shown by replacing a FT sensor by a combination of an F-BSPM and a V-BSPM. The F-BSPM predicts the expected intrinsic measurements while the V-BSPM estimates the overall FT values. The difference between both determines the amount and direction of extrinsic FT values. As a result, the robotic arm is equipped with accurate FT sensing capabilities which outperform the integrated analytical firmware.

A main methodological contribution is the operationalization of the BSPM approach using data-driven machine learning techniques. During a training phase, the behavior is continuously executed while recording proprioceptive sensor readings. The training data acquired from these demonstrations represents ground truth about the behavior-specific sensory-motor integration of the robot, i.e., the influence of performed actions and environmental conditions on the perceived proprioception. This data acquisition procedure does not require expert knowledge about the particular robot platform, e.g., kinematic chains or mass distribution, which is a major advantage over analytical approaches. As a result, the practical applicability of the BSPM approach is increased also for non-experts. Furthermore, the effort spent by the user as well as the wear and tear of the robot are reduced by learning from only a small set of training examples.

Dimensionality reduction and sensor selection schemes were utilized to automatically identify the most beneficial proprioceptors from the wide variety of sensor readings. Confirming previous findings, the well-known PCA performs well for robotic movement data. However, PCA does not take into account correlations with the learning target. This shortcoming is overcome by various sensor selection algorithms which are based on information-theoretic measures. In particular, TE was found to be beneficial to measure temporal information transfer from proprioceptors to the learning target and vice versa. Here, the direction needs to be chosen with regard to the particular application. More precisely, predicting the information transfer from the proprioceptors to the learning target is beneficial when selecting proprioceptors which influence the learning target. In contrast, the information transfer from the learning target to the proprioceptors needs to be taken into account when selecting proprioceptors which are influenced by the learning target. Instead of calculating information transfer, MMI and CMI are used to measure the shared information between proprioceptors and the learning target. Here, an iterative selection procedure was presented which subsequently selects proprioceptors by their additional amount of information where CMI performs slightly better than MMI. This allows selecting a minimal subset of less redundant proprioceptors which contains complete knowledge about the learning target and was demonstrated for weight classification. Here, ten proprioceptors was identified to share more than 99% of mutual information with the fill level of a self-contained water extraction station.

The most relevant sensor readings are then utilized to learn the particular BSPM in a supervised fashion where a mapping is typically achieved by lazy or eager machine learning. The main advantage of utilizing lazy learning is that a minimum of user effort is required. At runtime, a behavior is compared to the training examples detecting the most similar sequence of proprioceptive measurements. The main challenges here are to generate accurate outputs from a small number of training examples and to ensure a constantly low computation time. In this thesis, a novel interpolation scheme which extracts the underlying nonlinear dynamics of cyclic movement data by utilizing methods from the field of fluid dynamics (DMD) was presented. As a result, a finely sampled database of training examples was generated from about one minute of training data. Furthermore, an algorithmic approach with reduced computational complexity and constantly low computation time was introduced. In contrast, eager learning techniques generate an abstract model where the acquired data can be erased after training. Here, neural networks such as classical RNNs up to modern LSTMs were found to scale best with the number of training examples. In particular, the training effort can be scaled by reducing the complexity of the topology or limiting the number of learning epochs. Here, a runtime output is generated in constantly low time where more complex recurrent architectures and deep learning algorithms were found to be promising to increase the precision of BSPMs.

The BSPM approach was applied to different robot platforms where a wide variety of robotic applications were enhanced with accurate force sensing capabilities. As a result, these robots are enabled to flexibly interact with their environment during human-robot collaboration, tool-usage, exploration of unstructured environments and human-robot interaction. In particular, changes in the stability of a walking humanoid robot were monitored and related to the amount and direction of human guidance commands, e.g., pushing or pulling on the transported object. Both the F-BSPM or I-BSPM approaches were demonstrated to successfully solve the task. In another scenario, a V-BSPM was applied to an industrial robot arm to identify the torque exerted by a custom wrench. Furthermore, the weight of an object was classified where a sequence of 60 proprioceptive measurements (about half a second) was required to meet a certain decision. Here, the precision of the utilized LSTM classifier outperforms the usage of a dedicated FT sensor. Finally, the F-BSPM and V-BSPM approach were combined to generate accurate force estimates for human-robot interaction during performing pick-and-place operations. The difference between the force estimates of both models allows detecting deviations which are triggered by collisions with the environment. Here, forces applied while the robot stands still are interpreted as *intended* interactions of the human and used to trigger robotic behavior, e.g., grasping and releasing of objects. In contrast, extrinsic forces during behavior execution are attributed to *unintended* collisions with the environment or the human interactant which is used to implement a reliable fault detector and emergency stop. The generated force estimates are more accurate than the analytical firmware, require no precise configuration and further distinguish between intrinsic and extrinsic forces.

Summarizing, the presented BSPMs equip robots with precise force estimation capa-

bilities which often outperform dedicated FT sensors. These BSPMs are applicable to arbitrary robot platforms where at least some proprioceptive measurements are available, e.g., joint angles and motor currents. The information-theoretic sensor selection increases the learning efficiency and robustness of the corresponding models while being applicable without the need of expert knowledge. These are major advantages as compared to analytical models and related data-driven approaches for robotic force estimation.

8.2 Future Directions

To achieve accurate force estimation capabilities the weight classification task introduced in Section 6.2 uses an LSTM architecture for learning a mapping between proprioceptive sensor readings and weight of the lifted object. As experienced during the conducted evaluation, a well-chosen LSTM configuration, e.g., block and input size, is required to achieve meaningful force estimates. Here, finding a good configuration could be automatized by utilizing a hyperparameter optimizer algorithm. Such algorithms repeat the learning procedure for different configurations and search for an optimal hyperparameter configuration. This was not addressed in this thesis since it usually requires a lot of computational performance and therefore would have reduced the applicability of the BSPM approach in many settings. However, assuming the availability of a high-performance computer system this is promising to further automatize the learning procedure.

In order to generate BSPMs, only supervised learning techniques were investigated in this thesis. Here, the proprioception model is trained with labeled input-output training data which is then utilized to generate an estimate for runtime measurements. In future work, a reinforcement learning BSPM approach could be considered for tasks where no exact solution about the particular output is available. For this, the user may continuously reward the proprioceptive model with regard to the precision of the generated estimates. Involving such direct user feedback is independent from labeling training data and could allow the user to intuitively stop the learning procedure, i.e., when satisfied with the precision of the results or if no improvement is noticeable anymore.

Major contributions of this thesis are the selection methods of the most relevant sensors for the specific behavior. The presented selection schemes utilize TE and CMI which cannot distinguish between correlated and spurious correlated information. This problem was addressed by acquiring a sufficient amount of heterogeneous training data or monitoring the particular learning procedure with an additional set of validation data. The latter procedure could also be applied to the sensor selection step where the measured correlations are compared to unseen data. In this way, spurious correlations could be detected in an earlier step of the BSPM approach to prevent a time-consuming learning procedure with erroneous training data.

Thus far, the BSPM approach is applied from scratch for each new behavior. In future research, several BSPMs may be combined in a mixture-of-experts approach. This

would allow generalizing force estimation to new, unseen behaviors from a combination of similar ones. For this, it is important to implement a lifelong learning procedure which subsequently increases the database and consequently the number and precision of BSPMs.

The presented applications measure the overall extrinsic forces (e.g., the whole body during a cooperative transportation task) or were learned for behavior-specific contact points (e.g., the TCP while utilizing a custom torque wrench). Another interesting field of future work is to extend collision detection for a localization of environmental contact points. With some expert-knowledge about the kinematic structure and sensor positions various BSPMs could be learned hierarchically for different body parts. Hence, external forces could be estimated for each body part separately which would allow to estimate the contact point with the environment. This could also reduce the reaction time since the model could focus on a smaller set of sensors which are related to the particular body part. Finally, such a hierarchical structure would allow to apply the BSPM approach to swarms of robots which need to estimate forces for different environmental conditions. For example, this would allow carrying an object together while the swarm behavior is adapted with regard to the object's mass, the terrain and the transportation goal.

8.3 Concluding Remarks

Humans are able to recognize and compensate external forces by familiarizing themselves with task-specific proprioceptive feedback, e.g., muscle tensions. In contrast, robots are usually programmed by experts who have a precise analytical understanding of the task-specific dynamics. To overcome these divergences, this thesis introduced a novel machine learning concept which enhance robots with Behavior-Specific Proprioception Models (BSPMs). These models utilize data gathered from previous behavior demonstrations to estimate proprioceptive sensations during runtime. Differences between estimated and actually measured sensations are then utilized to estimate the amount and direction of extrinsic forces. As shown by various applications, the BSPM approach provides accurate force sensing capabilities for robots that do not have dedicated FT sensors.

Moreover, also robots with dedicated FT sensors may benefit from the proposed approach. Here, BSPMs could be utilized as an additional safeguard and failure protection. For example, when the real FT sensor is defective the virtual one could be used as fallback solution. In industrial settings, this would allow to maintain the production chain until a technician arrives to repair the hardware. The additional possibilities and application domains arising from the usage of BSPMs are manifold.

Bibliography

[1] Erik Berger, David Vogt, Christian Pönisch, Heni Ben Amor, and Bernhard Jung. *Cooperative human-robot manipulation tasks*. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Beyond Robot Grasping - Modern Approaches for Learning Dynamic Manipulation, 2012.

[2] Erik Berger, David Vogt, Heni Ben Amor, and Bernhard Jung. *Behavior adaptation in cooperative human-robot transportation tasks*. IEEE International Conference on Robotics, Automation (ICRA), Workshop on Semantics, Identification, and Control of Robot-Human-Environment Interaction, 2013.

[3] Erik Berger, David Vogt, Nooshin Haji-Ghassemi, Bernhard Jung, and Heni Ben Amor. "Inferring guidance information in cooperative human-robot tasks". In: *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*. IEEE. 2013, pp. 124–129.

[4] Erik Berger, Mark Sastuba, David Vogt, Bernhard Jung, and Heni Ben Amor. "Dynamic mode decomposition for perturbation estimation in human robot interaction". In: *Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on*. IEEE. 2014, pp. 593–600.

[5] Erik Berger, David Müller, David Vogt, Bernhard Jung, and Heni Ben Amor. "Transfer entropy for feature extraction in physical human-robot interaction: Detecting perturbations from low-cost sensors". In: *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE. 2014, pp. 829–834.

[6] Erik Berger, Mark Sastuba, David Vogt, Bernhard Jung, and Heni Ben Amor. "Estimation of perturbations in robotic behavior using dynamic mode decomposition". In: *Advanced Robotics* 29.5 (2015), pp. 331–343.

[7] Erik Berger, Steve Grehl, David Vogt, Bernhard Jung, and Heni Ben Amor. *Learning to estimate forces for safe tool usage*. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Safety for Human-Robot Interaction in Industrial Settings, 2015.

[8] Erik Berger, Steve Grehl, David Vogt, Bernhard Jung, and Heni Ben Amor. "Experience-based torque estimation for an industrial robot". In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 144–149.

[9] Erik Berger, David Vogt, Steve Grehl, Bernhard Jung, and Heni Ben Amor. "Estimating perturbations from experience using neural networks and information transfer". In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 176–181.

[10] New World Encyclopedia. *Proprioception — New World Encyclopedia,* [Online; accessed 24-April-2018]. 2008. URL: `http://www.newworldencyclopedia.org/p/index.php?title=Proprioception&oldid=682500`.

[11] Regina Yando, Victoria Seitz, and Edward Zigler. *Imitation: A developmental perspective.* Lawrence Erlbaum, 1978.

[12] Albert Bandura and Richard H Walters. "Social learning and personality development." In: (1963).

[13] Stefan Schaal. "Is imitation learning the route to humanoid robots?" In: *Trends in cognitive sciences* 3.6 (1999), pp. 233–242.

[14] Heni Ben Amor, Erik Berger, David Vogt, and Bernhard Jung. "Kinesthetic bootstrapping: Teaching motor skills to humanoid robots through physical interaction". In: *KI 2009: Advances in Artificial Intelligence*. Springer Berlin Heidelberg, 2009, pp. 492–499.

[15] Rolf Pfeifer and Christian Scheier. *Understanding intelligence*. MIT press, 2001.

[16] Kenneth Anderson, Lois E Anderson, and Walter D Glanze. *Mosby's medical, nursing, & allied health dictionary*. CV Mosby, 1998.

[17] Charles Sherrington. *The integrative action of the nervous system*. CUP Archive, 1910.

[18] PBC Matthews. "Muscle spindles and their motor control". In: *Physiological Reviews* 44.2 (1964), pp. 219–288.

[19] Léna Jami. "Golgi tendon organs in mammalian skeletal muscle: functional properties and central actions". In: *Physiological Reviews* 72.3 (1992), pp. 623–666.

[20] JAB Gray and M Sato. "Properties of the receptor potential in Pacinian corpuscles". In: *The Journal of physiology* 122.3 (1953), pp. 610–636.

[21] J.G. Steinbuch. *Beytrag zur Physiologie der Sinne*. Schrag, 1811.

[22] Charles Bell and Alexander Shaw. “Reprint of the ”Idea of a New Anatomy of the Brain,” with Letters, &c”. In: *J Anat Physiol* 3.Pt 1 (Nov. 1868). 17230788[pmid], pp. 147–182.

[23] H. Von Helmholtz. *Handbuch der Physiologischen Optik*. Ed. by von Helmholtz, H. 1867.

[24] E. von Holst and H. Mittelstaedt. “Das Reafferenzprinzip: Wechselwirkungen zwischen Zentralnervensystem und Peripherie”. In: *Naturwissenschaften* 37 (1950), pp. 464–476.

[25] Wen-Hua Chen, Jun Yang, Lei Guo, and Shihua Li. “Disturbance-observer-based control and related methods–An overview”. In: *IEEE Transactions on Industrial Electronics* 63.2 (2016), pp. 1083–1095.

[26] R. W. Sperry. “Neural basis of the spontaneous optokinetic response produced by visual inversion.” In: *Journal of comparative and physiological psychology* 43.6 (Dec. 1950), pp. 482–489. ISSN: 0021-9940.

[27] Mitsuo Kawato. “Internal models for motor control and trajectory planning”. In: *Current opinion in neurobiology* 9.6 (1999), pp. 718–727.

[28] Eugenia M Kolasinski. *Simulator Sickness in Virtual Environments.* Tech. rep. DTIC Document, 1995.

[29] Jaime G Carbonell, Ryszard S Michalski, and Tom M Mitchell. “An overview of machine learning”. In: *Machine learning*. Springer, 1983, pp. 3–23.

[30] Andres Munoz. “Machine Learning and Optimization”. In: *URL: https://www. cims. nyu. edu/˜ munoz/files/ml_optimization. pdf [accessed 2016-03-02][WebCite Cache ID 6fiLfZvnG]* (2014).

[31] Xin Li and Hsinchun Chen. “Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach”. In: *Decision Support Systems* 54.2 (2013), pp. 880–890.

[32] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. “End to end learning for self-driving cars”. In: *arXiv preprint arXiv:1604.07316* (2016).

[33] Sajjad Ahmad, Ajay Kalra, and Haroon Stephen. “Estimating soil moisture using remote sensing data: A machine learning approach”. In: *Advances in Water Resources* 33.1 (2010), pp. 69–80.

[34] Chintan Parmar, Patrick Grossmann, Johan Bussink, Philippe Lambin, and Hugo JWL Aerts. "Machine learning methods for quantitative radiomic biomarkers". In: *Scientific reports* 5 (2015).

[35] Khalil Benmouiza and Ali Cheknane. "Forecasting hourly global solar radiation using hybrid k-means and nonlinear autoregressive neural network models". In: *Energy Conversion and Management* 75 (2013), pp. 561–569.

[36] Pierre Lison. *An introduction to machine learning*. 2015.

[37] David W Aha. "Editorial". In: *Lazy learning*. Springer, 1997, pp. 7–10.

[38] Clarence W. Rowley, I. Igor Mez, I. Shervin Bagher, R. Philipp Schlatte, and Dan S. Henningson. "Spectral analysis of nonlinear flows". In: *Journal of Fluid Mechanics* 641.-1 (2009), pp. 115–127.

[39] Luai Al Shalabi and Zyad Shaaban. "Normalization as a preprocessing engine for data mining and the approach of preference matrix". In: *Dependability of Computer Systems, 2006. DepCos-RELCOMEX'06. International Conference on*. IEEE. 2006, pp. 207–214.

[40] Richard J Larsen, Morris L Marx, et al. *An introduction to mathematical statistics and its applications*. Vol. 3. Prentice-Hall Englewood Cliffs, NJ, 2000, p. 282.

[41] Katerina Hlavkov-Schindler, Milan Palu, Martin Vejmelka, and Joydeep Bhattacharya. "Causality detection based on information-theoretic approaches in time series analysis". In: *Physics Reports* 441.1 (2007), pp. 1 –46. ISSN: 0370-1573.

[42] Kenneth J Hintz. "A measure of the information gain attributable to cueing". In: *IEEE Transactions on Systems, Man, and Cybernetics* 21.2 (1991), pp. 434–442.

[43] James Manyika and Hugh Durrant-Whyte. *Data Fusion and Sensor Management: a decentralized information-theoretic approach*. Prentice Hall PTR, 1995.

[44] Juan Liu, James Reich, and Feng Zhao. "Collaborative in-network processing for target tracking". In: *EURASIP Journal on Advances in Signal Processing* 2003.4 (2003), p. 616720.

[45] Amaury Dame and Eric Marchand. "Mutual information-based visual servoing". In: *IEEE Transactions on Robotics* 27.5 (2011), pp. 958–969.

[46] Matei Ciocarlie, Corey Goldfeder, and Peter Allen. “Dimensionality reduction for hand-independent dexterous robotic grasping”. In: *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE. 2007, pp. 3270–3275.

[47] Panagiotis K Artemiadis and Kostas J Kyriakopoulos. “EMG-based control of a robot arm using low-dimensional embeddings”. In: *IEEE Transactions on Robotics* 26.2 (2010), pp. 393–398.

[48] Sebastian Bitzer, Matthew Howard, and Sethu Vijayakumar. “Using dimensionality reduction to exploit constraints in reinforcement learning”. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE. 2010, pp. 3219–3225.

[49] Geoffrey E Hinton and Ruslan R Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *science* 313.5786 (2006), pp. 504–507.

[50] Heni Ben Amor. “Imitation learning of motor skills for synthetic humanoids”. PhD thesis. Technische Universität Bergakademie Freiberg, 2010.

[51] Sami Haddadin, Alessandro De Luca, and Alin Albu-Schäffer. “Robot collisions: A survey on detection, isolation, and identification”. In: *IEEE Transactions on Robotics* 33.6 (2017), pp. 1292–1312.

[52] Andreas Stolt, Magnus Linderoth, Anders Robertsson, and Rolf Johansson. “Force controlled robotic assembly without a force sensor”. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 1538–1543.

[53] Sebastian Erhart, Dominik Sieber, and Sandra Hirche. “An impedance-based control architecture for multi-robot cooperative dual-arm mobile manipulation”. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 315–322.

[54] Arne Wahrburg, Anders Robertsson, Björn Matthias, Fan Dai, and Hao Ding. “Improving contact force estimation accuracy by optimal redundancy resolution”. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 3735–3741.

[55] Adria Colomé, Diego Pardo, Guillem Alenya, and Carme Torras. “External force estimation during compliant robot manipulation”. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 3535–3540.

[56] Sethu Vijayakumar and Stefan Schaal. "Locally weighted projection regression: An O (n) algorithm for incremental real time learning in high dimensional space". In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*. Vol. 1. 2000, pp. 288–293.

[57] Wei He, Shuzhi Sam Ge, Yanan Li, Effie Chew, and Yee Sien Ng. "Neural network control of a rehabilitation robot by state and output feedback". In: *Journal of Intelligent & Robotic Systems* 80.1 (2015), pp. 15–31.

[58] Johannes Schröder-Schetelig, Poramate Manoonpong, and Florentin Wörgötter. "Using efference copy and a forward internal model for adaptive biped walking". In: *Autonomous Robots* 29.3 (2010), pp. 357–366.

[59] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[60] Frank Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[61] John C Hay, FC Martin, and CW Wightman. "The mark-1 perceptron-design and performance". In: *Proceedings of the institute of radio engineers*. Vol. 48. 3. 1960, pp. 398–399.

[62] Marvin Minsky and Seymour Papert. "Perceptrons." In: (1969).

[63] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), p. 533.

[64] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[65] Gioqinang Zhang and Michael Y Hu. "Neural network forecasting of the British pound/US dollar exchange rate". In: *Omega* 26.4 (1998), pp. 495–506.

[66] Zainab H Osman, Mohamed L Awad, and Tawfik K Mahmoud. "Neural network based approach for short-term load forecasting". In: *Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES*. IEEE. 2009, pp. 1–8.

[67] Ashif Panakkat and Hojjat Adeli. "Recurrent neural network for approximate earthquake time and location prediction using multiple seismicity indicators". In: *Computer-Aided Civil and Infrastructure Engineering* 24.4 (2009), pp. 280–292.

[68] FW Lewis, Suresh Jagannathan, and A Yesildirak. *Neural network control of robot manipulators and non-linear systems*. CRC Press, 1998.

[69] Jasmin Velagic, Nedim Osmic, and Bakir Lacevic. “Neural network controller for mobile robot motion control”. In: *World Academy of Science, Engineering and Technology* 47 (2008), pp. 193–198.

[70] Ching-Chih Tsai, Hsu-Chih Huang, and Shui-Chun Lin. “Adaptive neural network control of a self-balancing two-wheeled scooter”. In: *IEEE Transactions on Industrial Electronics* 57.4 (2010), pp. 1420–1428.

[71] Naveen Kumar, Vikas Panwar, Nagarajan Sukavanam, Shri Prakash Sharma, and Jin-Hwan Borm. “Neural network based hybrid force/position control for robot manipulators”. In: *International Journal of Precision Engineering and Manufacturing* 12.3 (2011), pp. 419–426.

[72] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. “Rotation invariant neural network-based face detection”. In: *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*. IEEE. 1998, pp. 38–44.

[73] Paul Viola and Michael Jones. “Rapid object detection using a boosted cascade of simple features”. In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2001, pp. I–I.

[74] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. “Imagenet: A large-scale hierarchical image database”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 248–255.

[75] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[76] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[77] Phil Kim. “Convolutional Neural Network”. In: *MATLAB Deep Learning*. Springer, 2017, pp. 121–147.

[78] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[79] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[80] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. "Large scale distributed deep networks". In: *Advances in neural information processing systems*. 2012, pp. 1223–1231.

[81] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[82] Ronen Eldan and Ohad Shamir. "The power of depth for feedforward neural networks". In: *Conference on Learning Theory*. 2016, pp. 907–940.

[83] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667.

[84] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Training very deep networks". In: *Advances in neural information processing systems*. 2015, pp. 2377–2385.

[85] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. "Sequence to sequence-video to text". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4534–4542.

[86] Douglas Eck and Juergen Schmidhuber. "Finding temporal structure in music: Blues improvisation with LSTM recurrent networks". In: *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*. IEEE. 2002, pp. 747–756.

[87] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. "LSTM neural networks for language modeling". In: *Thirteenth Annual Conference of the International Speech Communication Association*. 2012.

[88] Alexander Förster, Alex Graves, and Jürgen Schmidhuber. "RNN-based Learning of Compact Maps for Efficient Robot Localization." In: *ESANN*. 2007, pp. 537–542.

[89] Dickson Neoh Tze How, Khairul Salleh Mohamed Sahari, Hu Yuhuang, and Loo Chu Kiong. "Multiple sequence behavior recognition on humanoid robot using long short-term memory (LSTM)". In: *Robotics and Manufacturing Automation (ROMA), 2014 IEEE International Symposium on*. IEEE. 2014, pp. 109–114.

[90] Sebastian Otte, Christian Weiss, Tobias Scherer, and Andreas Zell. “Recurrent Neural Networks for fast and robust vibration-based ground classification on mobile robots”. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 5603–5608.

[91] Matthias Luber, Luciano Spinello, Jens Silva, and Kai O Arras. “Socially-aware robot navigation: A learning approach”. In: *Intelligent robots and systems (IROS), 2012 IEEE/RSJ international conference on*. IEEE. 2012, pp. 902–907.

[92] Claude Shannon. “A Mathematical Theory of Communication”. In: *Bell System Technical Journal* 27 (1948), pp. 379–423, 623–656.

[93] Thomas Schreiber. “Measuring Information Transfer”. In: *Phys. Rev. Lett.* 85 (2 July 2000), pp. 461–464.

[94] Shinya Ito, Michael E. Hansen, Randy Heiland, Andrew Lumsdaine, Alan M. Litke, and John M. Beggs. “Extending Transfer Entropy Improves Identification of Effective Connectivity in a Spiking Cortical Network Model”. In: *PLoS ONE* 6.11 (Nov. 2011), pp. 1–13.

[95] Alessandro Montalto, Luca Faes, and Daniele Marinazzo. “MuTE: A MATLAB Toolbox to Compare Established and Novel Estimators of the Multivariate Transfer Entropy”. In: *PLoS ONE* 9.10 (Oct. 2014), pp. 1–13.

[96] Frederico A.C. Azevedo, Ludmila R.B. Carvalho, Lea T. Grinberg, Jos Marcelo Farfel, Renata E.L. Ferretti, Renata E.P. Leite, Wilson Jacob Filho, Roberto Lent, and Suzana Herculano-Houzel. “Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain”. In: *The Journal of Comparative Neurology* 513.5 (2009), pp. 532–541. ISSN: 1096-9861.

[97] David Kriesel. *A Brief Introduction to Neural Networks*. 2007.

[98] B. Widrow and M. E. Hoff. “Adaptive Switching Circuits”. In: *Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, Part 4*. 1960, pp. 96–104.

[99] R. O. Winder. “Single stage threshold logic”. In: *2nd Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1961)*. Oct. 1961, pp. 321–332.

[100] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314. ISSN: 1435-568X.

[101] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[102] Paul J Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[103] S. Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München*. 1991.

[104] F. A. Gers, J. Schmidhuber, and F. Cummins. "Learning to Forget: Continual Prediction with LSTM". In: *Neural Computation* 12 (2000), pp. 2451–2471.

[105] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. "LSTM: A search space odyssey". In: *IEEE transactions on neural networks and learning systems* (2016).

[106] Hiroaki Sakoe. "Dynamic programming algorithm optimization for spoken word recognition". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26 (1978), pp. 43–49.

[107] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. "A unifying view of sparse approximate Gaussian process regression". In: *Journal of Machine Learning Research* 6.Dec (2005), pp. 1939–1959.

[108] W Ross Ashby. *An introduction to cybernetics*. Chapman & Hall Ltd, 1961.

[109] William D Smart. "Is a common middleware for robotics possible?" In: *Proceedings of the IROS 2007 workshop on Measures and Procedures for the Evaluation of Robot Architectures and Middleware. Citeseer*. Vol. 1. 2007.

[110] Hubert P. H. Shum, Taku Komura, Takaaki Shiratori, and Shu Takagi. "Physically-Based Character Control in Low Dimensional Space". In: *Motion in Games: Third International Conference, MIG 2010, Utrecht, The Netherlands, November 14-16, 2010. Proceedings*. Ed. by Ronan Boulic, Yiorgos Chrysanthou, and Taku Komura. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 23–34. ISBN: 978-3-642-16958-8.

[111] Sergey Levine, Jack M. Wang, Alexis Haraux, Zoran Popović, and Vladlen Koltun. "Continuous Character Control with Low-Dimensional Embeddings". In: *ACM Transactions on Graphics* 31.4 (2012), p. 28.

[112] I. T. Jolliffe. *Principal Component Analysis*. Second. Springer, 2002.

[113] Sam T. Roweis and Lawrence K. Saul. “Nonlinear dimensionality reduction by locally linear embedding”. In: *Science* 290 (2000), pp. 2323–2326.

[114] Matthew Brand and Matthew Brand. “Charting a Manifold”. In: *Advances in Neural Information Processing Systems 15*. MIT Press, 2003, pp. 961–968.

[115] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*. Vol. 1. MIT press Cambridge, 2006.

[116] Edward Snelson and Zoubin Ghahramani. “Sparse Gaussian Processes using Pseudo-inputs”. In: *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*. MIT press, 2006, pp. 1257–1264.

[117] Neil D. Lawrence, John C. Platt, and Michael I. Jordan. “Extensions of the Informative Vector Machine”. In: *Proceedings of the First International Conference on Deterministic and Statistical Methods in Machine Learning*. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 56–87.

[118] Peter J. Schmid. “Dynamic mode decomposition of numerical and experimental data”. In: *Journal of Fluid Mechanics* 656 (Aug. 2010), pp. 5–28. ISSN: 1469-7645.

[119] Shervin Bagheri. “Analysis and control of transitional shear flows using global modes”. PhD thesis. KTH, Mechanics, 2010, pp. viii, 82.

[120] KevinK. Chen, JonathanH. Tu, and ClarenceW. Rowley. “Variants of Dynamic Mode Decomposition: Boundary Condition, Koopman, and Fourier Analyses”. English. In: *Journal of Nonlinear Science* 22.6 (2012), pp. 887–915. ISSN: 0938-8974.

[121] Mihailo R. Jovanovi, Peter J. Schmid, and Joseph W. Nichols. “Sparsity-promoting dynamic mode decomposition”. In: *Physics of Fluids (1994-present)* 26.2, 024103 (2014).

[122] Meinard Müller. *Information Retrieval for Music and Motion*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007, pp. 79–84. ISBN: 3540740473.

[123] J. Brooke. “SUS: A quick and dirty usability scale”. In: *Usability evaluation in industry*. London: Taylor and Francis, 1996.

[124] Aaron Bangor, Philip T. Kortum, and James T. Miller. “An Empirical Evaluation of the System Usability Scale”. In: *International Journal of Human-Computer Interaction* 24.6 (2008), pp. 574–594.

[125] Claudia Buhl, Marc Donner, Marvin Ferber, Steve Grehl, Marco Herrmann, Bernhard Jung, and Peter Poschmann. “Mobile Roboter für Mapping und Monitoring im Bergbau”. In: *Proceedings 16. Geokinematischen Tag*. 2015.

[126] Steve Grehl, Marc Donner, Marvin Ferber, Anne Dietze, Helmut Mischo, and Bernhard Jung. “Mining-RoX Mobile Robots in Underground Mining”. In: *Proceedings Third International Future Mining Conference*. AUSIMM. 2015, 5764.

[127] Steve Grehl, Helmut Mischo, Bernhard Jung, et al. “Research perspective-mobile robots in underground mining”. In: *AusIMM Bulletin* Feb 2017 (2017), p. 44.

[128] Johannes Gebel. “Erarbeitung eines Konzeptes zur robotergestützten Entnahme von Wasserproben (Development of a concept for the robot-aided taking of water samples) n”. In: *Project-Thesis* (2016), Freiberg University of Mining and Technology.

[129] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. New York: Springer, 2007, p. 198. ISBN: 978-0-387-31073-2.

[130] Jeff Heaton. *Introduction to neural networks with Java*. Heaton Research, Inc., 2008.

[131] Kevin J. Lang, Alex H. Waibel, and Geoffrey E. Hinton. “A time-delay neural network architecture for isolated word recognition”. In: *Neural Networks* 3.1 (1990), pp. 23 –43. ISSN: 0893-6080.

[132] Tsungnan Lin, B. G. Horne, P. Tino, and C. L. Giles. “Learning Long-term Dependencies in NARX Recurrent Neural Networks”. In: *Trans. Neur. Netw.* 7.6 (Nov. 1996), pp. 1329–1338. ISSN: 1045-9227.

[133] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion”. In: *J. Mach. Learn. Res.* 11 (Dec. 2010), pp. 3371–3408. ISSN: 1532-4435.